Reports Rec'd Posted 8/15

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF RESEARCH ADMINISTRATION

RESEARCH PROJECT INITIATION

Date: July 11, 1973

Project Title: Quadratic Set Covering Problem

Project No: E-24-618

Principal Investigator: Dr. M. S. Bazaraa

Sponsor: National Science Foundation

Agreement Period: From July 1, 1973 Until February 28, 1975*

Type Agreement: Grant No. GK-38337

Amount: $30,700 NSF (E-24-618)
3,363 GIT Contrib. (E-24-315)
$34,063 TOTAL

Reports Required: Annual Technical Report; Final Technical Report.

Sponsor Contact Person (s): Technical Matters

Dr. M. S. Ghausi, Head
Electrical Sciences and Analysis Section
System Theory and Applications Program
National Science Foundation
Washington, D.C. 20550
Phone: (202) 632-5881

Administrative Matters
Mr. Wilbur W. Bolton, Jr.
Grants Officer
National Science Foundatio
Washington, D.C. 20550

*All commitments to be met by this date unless formal extension of the grant is obtained in advance. Basic work period (14 mos.) ends August 31, 1974.

Assigned to: ~~School of I&SE~~

COPIES TO:

Principal Investigator	Library
School Director	Rich Electronic Computer Center
Dean of the College	Photographic Laboratory
Director, Research Administration	Project File
Director, Financial Affairs (2)	
Security-Reports-Property Office ✓	
Patent Coordinator	Other

RA-3 (6-71)

Pos BL
Acct
OHL

GEORGIA INSTITUTE OF TECHNOLOGY

OFFICE OF RESEARCH ADMINISTRATION

RESEARCH PROJECT TERMINATION

Date: March 13, 1975

Project Title: Quadratic Set Covering Problem

Project No: E-24-618

Principal Investigator: Dr. M. S. Bazaraa

Sponsor: National Science Foundation

Effective Termination Date: 2/28/75

Clearance of Accounting Charges: 2/28/75

Grant/Contract Closeout Actions Remaining: ~~XXXX~~ None

Assigned to School of: Industrial and Systems Engineering

COPIES TO:

- Principal Investigator
- School Director
- Dean of the College
- Director of Research Administration
- Office of Financial Affairs (2)
- Security - Reports - Property Office
- Patent and Inventions Coordinator
- Library, Technical Reports Section
- Computer Sciences
- Photographic Laboratory
- Terminated Project File No. _____
- Other _____

(5/70)

THE QUADRATIC SET COVERING (ASSIGNMENT) PROBLEM: APPLICATIONS AND COMPUTATION

Annual Report

June 1974

NSF Grant # GK - 38337

THE QUADRATIC SET COVERING (ASSIGNMENT) PROBLEM: APPLICATIONS AND COMPUTATION

Summary

Linear set covering and set partitioning problems have attracted the attention of many researchers. This is due to the simple formulation of these problems and their wide ranges of applications. Even though the formulations of these problems is quite simple, solving them is challenging due to the difficulties involved with the 0-1 nature of the variables.

This research concerns itself with quadratic set covering and set partitioning problems. The significance of these problems is reflected by the flexibility of the quadratic objective, and its ability to deal with first order interaction between the variables, which leads to a wider range of applications. This report includes three sections. In the first section various potential applications of the quadratic set covering and quadratic assignment problems are presented. The second section deals with a branch and bound algorithm for obtaining optimal and suboptimal solutions of the quadratic assignment problem. In the last section we develop a branch and bound algorithm for solving the special case of the traveling salesman problem (symmetric and nonsymmetric). A pre-branch and bound dual optimization based on finding minimal 1-trees is used in order to speed the tree search.

1. The Quadratic Set Covering Problem, Related Problems, and Applications

1.1 Mathematical Descriptions of Nine Related Problems

1.2 The Location Problems
- 1.21 Facilities Location
- 1.22 Building Layout
- 1.23 Central Panel Layout
- 1.24 Wiring Problem

1.3 The Traveling Salesman Problem
- 1.31 The Classical Traveling Saleman Problem
- 1.32 Four Modifications of the Traveling Salesman Problem
- 1.33 An Application to Production Scheduling
- 1.34 The Multi-Traveling Salesman Problem, Its Generalizations, and Some Applications

1.4 Applications of the QSCP and Related Problems
- 1.41 Routing Problems: Truck Delivery and School Bussing
- 1.42 Location of Fire Hydrants
- 1.43 Airline Crew Scheduling
- 1.44 Assembly Line Balancing
- 1.45 Information Retrieval
- 1.46 Attack/Defense Networks
- 1.47 Project Selection and Capital Budgeting
- 1.48 Political Districting

1.5 Miscellaneous Applications

1.6 Some Comments on the Practicability of QSCP Applications

Section 1 The Quadratic Set Covering Problem, Related Problems and Applications

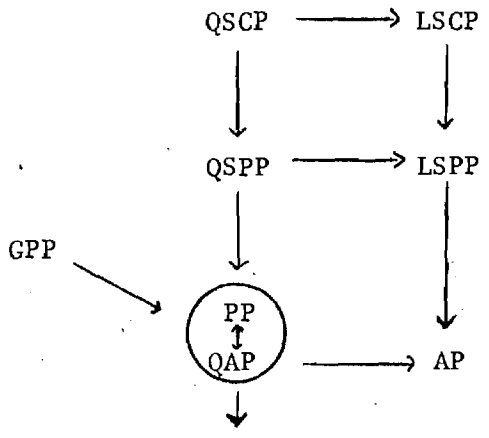
In this section we describe the quadratic set covering problem and eight other related problems, and give a number of examples illustrating the applications of these problems. Attention is given to the relationship between the problems, and a new generalized problem is presented with the threefold purpose of unifying several of the other problems, modelling more complicated extensions of the applications, and providing for more efficient solutions in some instances. Some examples are straightforward applications, while others serve to show how some applications are better modelled by quadratic than linear problems. One example points out the inadequacy of the quadratic set covering model, and indications are given for the development of a new problem. In some examples we point out the analogous factors in apparently diverse applications and identify complications unique to each application. Some new extensions of known applications are also given. [Garfinkel and Neurhanser (1972), Ch. 8]

1.1 Mathematical Descriptions of Nine Related Problems

We shall consider the following problems:

quadratic set covering problem	QSCP
quadratic set partitioning problem	QSPP
linear set covering problem	LSCP
linear set partitioning problem	LSPP
generalized placement problem (new)	GPP
placement problem	PP
quadratic assignment problem	QAP
assignment problem	AP
traveling salesman problem	TSP

All of the problems, except the GPP which is new, are well known. As the problems are precisely described below, it will be helpful to keep in mind the following diagram showing the relationships of the problems as special cases of the others in the formulations we give.



It should be noted that the mathematical formulations of the PP and QAP are identical, although the PP appears superficially to be more general. Hence, they are considered as a single problem here.

Let x be an n x 1 binary vector, x' the transpose of x, A an m x n matrix of zeros and ones, C an n x n matrix, and e_m an m x 1 vector of ones. The quadratic set covering problem (QSCP) is

$$\begin{aligned} &\text{Minimize} && x'Cx \\ &\text{such that} && Ax \geq e_m \\ & && x \text{ is a binary vector} \end{aligned}$$

and the quadratic set partitioning problem (QSPP) is

$$\begin{aligned} &\text{Minimize} && x'Cx \\ &\text{such that} && Ax = e_m \\ & && x \text{ is a binary vector} \end{aligned}$$

If we let I_n be the n x n identity matrix, then by choosing A to be the 2n x n^2 matrix

$$A = \begin{bmatrix} e' & & & \\ & e' & & \\ & & \ddots & \\ & & & e' \\ I_n & I_n & \cdots & I_n \end{bmatrix}$$

we obtain the quadratic assignment problem (QAP) for the case of assigning n objects to n locations. We may always assume, without loss of generality, that the number of objects to be assigned is equal to the number of locations, for if this is not the case, then we may add dummy objects which make them equal but have no effect on the final solution. By replacing the objective functions of the QSCP, QSPP, and QAP with a linear, instead of quadratic function, we obtain the linear set covering problem (LSCP), linear set partitioning problem (LSPP), and assignment problem (AP) respectively. The placement problem (PP) is simply the QAP with a linear expression $d'x$ added to the objective function (here, d is an 1 x n cost vector), but the 0-1 variables x_i permit this linear expression to be absorbed into the quadratic portion of the objective function by adding the elements of d to the diagonal of C. Hence, from our viewpoint the PP and QAP are the same, although it may appear that the PP is more complicated when one is looking at real applications. The traveling

salesman problem (TSP), the problem of a salesman who must visit n cities, each exactly once, and return home while minimizing the total distance of his trip will be formulated later as a special case of the QAP. The generalized placement problem (GPP) includes the PP and QAP as special cases, but allows one to require that certain sets of objects must be assigned to certain sets of locations, or certain sets of locations must be filled by objects from certain sets of objects (Geddes 1974). As will be seen in the applications, these are very realistic restrictions, and when the GPP is formulated in terms of 0-1 variables the complications are actually helpful in obtaining the solution more efficiently. Before describing the GPP we need to introduce some notation. For a finite set S let $|S|$ = the number of elements of S and $P(S) = \{X | X \subset S\}$ = the power set of S. Suppose we are given two finite sets A and B with $|A| = |B|$. (Since A is thought of as a set of elements to be assigned to locations B, we must have $|A| \leq |B|$. There is no loss generality in assuming $|A| = |B|$, since "dummy" elements may always be added to A which do not influence the objective function.) $\Phi = \{\psi | \psi: A \rightarrow B, \ \psi \text{ 1-1 map}\}$. Assume that fixed assignment costs of elements of A to B, quantities shipped, and costs of shipping have been taken into consideration to produce a quadratic objective function, as in the QAP, and let c_ψ be the cost associated with $\psi \in \Phi$. Let $R \subset P(A) \times P(B)$. We say that an element $\psi \in \Phi$ satisfies R, written ψSR, provided:

$$\text{for each } (\hat{A}, \hat{B}) \in R, \qquad \begin{aligned} \psi[\hat{A}] &\subset \hat{B} \quad \text{if} \quad |\hat{A}| < |\hat{B}| \\ \psi^{-1}[\hat{B}] &\subset \hat{A} \quad \text{if} \quad |\hat{A}| > |\hat{B}| \\ \psi[\hat{A}] &= \hat{B} \quad \text{if} \quad |\hat{A}| = |\hat{B}| \end{aligned}$$

Here $\psi[\hat{A}]$ denotes the image of the set $\hat{A}$ under the map ψ. The generalized placement problem (GPP) is

$$\begin{aligned} &\text{minimize} && c_\psi \\ &\text{subject to} && \psi \in \Phi \\ &\text{and} && \psi\text{SR} \end{aligned}$$

where A, B, R, and the costs c_ψ are given. Note that if $R = \{(A,B)\}$ we simply have the QAP. Other cases, such as when R is of the form $R = \{(\{a_{i_1}\}, \{b_{i_1}\}), \ldots, (\{a_{i_s}\}, \{b_{i_s}\})\}$, $s < |A|$, will be considered in sections 1.2 and 1.32, where examples are given for location problems and the TSP. A procedure is also given for writing the GPP in terms of 0-1 variables.

1.2 The Location Problems

Because of their similar mathematical structure as quadratic assignment problems, as well as analogous physical and economic interpretations, we give four applications under the general classification of location problems. These are

facilities location
building layout
control panel layout
wiring problem

Each application will be discussed in detail individually, but first the location problem in general, including properties and extensions common to all of the applications, is described. We suppose that n objects a_i $(i = 1,2,\ldots,n)$ are to be assigned to n locations b_k $(k = 1,2,\ldots,n)$ in such a way that each object is assigned to one and only one location. As was mentioned earlier, it is easily seen that if we have m objects and n locations with $m < n$, we may supply $n - m$ dummy objects so that there is no loss of generality in assuming the same number of objects and locations. There may or may not be a fixed cost f_{ik} associated with assigning object a_i to location b_k. There are given a fixed number of "units" to be "shipped" from object a_i to object a_j, say u_{ij}, and in general $u_{ij} \neq u_{ji}$. These units may be interpreted as trips made by people in the building layout problems, as we shall see, or as numbers of wire connections in the wiring problem. Shipping one unit from location b_k to b_ℓ incurs a cost $d_{k\ell}$ (usually related to distance). Define n^2 variables x_{ik} $(i = 1,2,\ldots,n;\ \ k = 1,2,\ldots,n)$ by

$$x_{ik} = \begin{cases} 1 & \text{if object } a_i \text{ is placed in location } b_k \\ 0 & \text{otherwise} \end{cases}$$

and let $x' = (x_{1,1}\ x_{1,2}\ \cdots\ x_{1,n}\ ;\ x_{n,1}\ x_{n,2}\ \cdots\ x_{n,n})$.

For $i = 1,2,\ldots,n$ and $j = 1,2,\ldots n$, define n^2 matrices of order $n \times n$

$$C_{ij} = \left(C \begin{matrix} ij \\ rs \end{matrix}\right) \qquad (r = 1,2,\ldots,n;\ \ s = 1,2,\ldots,n)$$

where, if $i = j$, then $C \begin{matrix} ij \\ rs \end{matrix} = \begin{cases} f_{ir} & \text{if } r = s \\ 0 & \text{if } r \neq s \end{cases}$

$$\text{if } i \neq j, \quad \text{then} \quad C_{rs}^{ij} = \begin{cases} 0 & \text{if} \quad r = s \\ u_{ij} d_{rs} & \text{if} \quad r \neq s \end{cases}$$

C_{rs}^{ij} is therefore interpreted as the cost of shipping u_{ij} units from object a_i to object a_j, if they were located in b_r and b_s respectively

Let C be the partitioned matrix

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & & & \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}$$

Let e_n be a n x 1 vector of ones and I_n be the n x n identity matrix.

Let H be the 2n x n^2 assignment matrix

$$H = \begin{bmatrix} e_n' & & & \\ & e_n' & & \\ & & \ddots & \\ & & & e_n' \\ I_n & I_n & \cdots & I_n \end{bmatrix}$$

The location problem (QAP, PP) is to

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_{2n}$$

$$x \text{ binary}$$

It should be noted that a large number of elements of the C matrix were set equal to zero since they correspond to assignments not permitted by the constraints $Hx = e_{2n}$. The first n rows of equations represented by $Hx = e_{2n}$ require that each object is assigned to one location, and the remaining n equations require that each location is assigned to one object; that is to say we have a one-to-one correspondence between objects and locations. Now, for example, the object a_1 cannot be assigned to locations b_1 and b_3 simultaneously, so the first row third column entry of

C_{11}, c_{13}^{11}, was set equal to zero. Actually, any number could be placed in this entry, since it would never contribute to the value of the objective functions $x'Cx$ for any feasible x.

There are two extensions of the location problem which fall into the case of the GPP. In one of these we may require a few selected objects to be assigned to particular locations. For example, we may be given a collection of factories and warehouses already built and wish to add some new ones. A slightly different problem occurs when certain sets of objects must be assigned to certain sets of locations, or certain sets of locations must be filled by objects from given sets. In the second instance, we still have some freedom to locate the object. An example is that the tuning knob on a radio has to be, say, on the outside in front and not buried in the middle of the set, but there are several locations available. Another example is given below in which ten plants of various sizes, some of which require a water supply, are to be located on ten sites of different sizes, some of which have water supplies. This type of constraint may also be used to keep certain objects either close together or far apart. In a warehouse certain chemicals cannot be stored together for safety reasons such as fire and explosion. Certain areas in a warehouse may be unsuitable for storing some materials because of temperature or humidity, while some materials may be stored in any area of the warehouse. We may wish to require certain rooms in an office building to be located on an outside wall. In the design of electrical equipment some components may fail to function properly if located too far apart, while others fail when located too close together. An alternative to using the GPP to handle these complications would be to add constraints which simply require the distance between the locations containing objects be greater than (or less than) some fixed number. This may allow for lower objective function values, but may also give a difficult problem to solve.

1.21 Facilities Location

We begin by describing a simple example in which three plants a_i ($i = 1,2,3$) are to be assigned to three locations b_k ($k = 1,2,3$). The fixed costs f_{ik} of assigning plant a_i to location b_k are given in Table 1. Table 2 gives the number of units u_{ij} to be shipped from plant a_i to plant a_j, and Table 3 gives the unit costs $d_{k\ell}$ of shipping from location b_k to b_ℓ.

Plant a_i \ Location b_k	1	2	3
1	200	300	250
2	400	350	300
3	600	700	650

Table 1. Values of f_{ik}

Plant a_i \ Plant a_j	1	2	3
1	-	50	100
2	60	-	80
3	40	70	-

Table 2. Values of u_{ij}

Location b_k \ Location b_1	1	2	3
1	-	2	3
2	1	-	2
3	3	3	-

Table 3. Values of $d_{k\ell}$

We compute the C and H matrices described above:

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

$$= \left[\begin{array}{ccc|ccc|ccc} 200 & - & - & - & 100 & 150 & - & 200 & 300 \\ - & 300 & - & 50 & - & 100 & 100 & - & 200 \\ - & - & 250 & 150 & 150 & - & 300 & 300 & - \\ \hline - & 120 & 180 & 400 & - & - & - & 160 & 240 \\ 60 & - & 120 & - & 350 & - & 80 & - & 160 \\ 180 & 180 & - & - & - & 300 & 240 & 240 & - \\ \hline - & 50 & 120 & - & 140 & 210 & 600 & - & - \\ 40 & - & 80 & 70 & - & 140 & - & 700 & - \\ 120 & 120 & - & 210 & 210 & - & - & - & 650 \end{array}\right]$$

$$H = \begin{bmatrix} 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & 1 & & & \\ & & & & & & 1 & 1 & 1 \\ 1 & & & 1 & & & 1 & & \\ & 1 & & & 1 & & & 1 & \\ & & 1 & & & 1 & & & 1 \end{bmatrix}$$

As before, for i = 1,2,3 ; k = 1,2,3 ; let

$$x_{ik} = \begin{cases} 1 & \text{if plant } a_i \text{ is assigned to location } b_k \\ 0 & \text{otherwise} \end{cases}$$

$$x' = (x_{11}\ x_{12}\ x_{13},\ x_{21}\ x_{22}\ x_{23},\ x_{31}\ x_{32}\ x_{33})$$

and $e_6' = (1\ 1\ 1\ 1\ 1\ 1)$.

The location problem is then

minimize $x'Cx$

subject to $Hx = e_6$

and x is a binary vector

It should be emphasized that in this problem the quantities shipped between plants, u_{ij}, were given and fixed, which in more general instances is not always the case. There are a number of complications which may be introduced. For example, the shipment of goods between plants, if placed in certain locations, may require the construction of a road or pipeline, and the fixed cost of such construction must be taken into consideration. It is also possible that more than one plant may be placed in one location. A very common problem is that we already have a number of plants built and wish to add a few more. Suppose we are considering n plants a_i and n locations b_k, but $s < n$ of these plants are already built. In particular, assume that for $j = 1,2,\ldots,s$ we have plant a_{i_j} built in location b_{i_j}. We can formulate this problem as a GPP by letting

$$R = \{(\{a_{i_1}\},\{b_{i_1}\}),\ (\{a_{i_2}\},\{b_{i_2}\}),\ldots,\ (\{a_{i_s}\},\{b_{i_s}\})\}$$

A second more complicated example of the GPP is given, and the relationship of the 0-1 variables shown. Let $A = \{a_1, a_2, \ldots, a_{10}\}$ be a set of ten plants and $B = \{b_1, b_2, \ldots, b_{10}\}$ be a set of ten locations. We are given that u_{ij} units are to be shipped from plant a_i to plant a_j, the cost of shipping one unit from location b_k to b_ℓ is $d_{k\ell}$, and the cost of placing plant a_i in location b_k is f_{ik}.

Let $A_1 = \{a_8, a_9, a_{10}\}$

$B_1 = \{b_8, b_9, b_{10}\}$

$A_2 = \{a_4, a_5, a_6, a_7, a_8\}$

$B_2 = \{b_1, b_2, b_3, b_4, b_5, b_8, b_9\}$

$A_3 = \{a_1, a_2, a_3, a_4, a_5, a_9\}$

and $B_3 = \{b_2, b_7, b_8\}$

Let $R = \{(A_1,B_1), (A_2,B_2), (A_3,B_3)\} \subset P(A) \times P(B)$

We define C and H as before and obtain the problem:

$$\text{minimize} \quad x'Cx \tag{1}$$

$$\text{subject to} \quad Hx = e_{20} \tag{2}$$

$$x \text{ binary} \tag{3}$$

and elements from A_1 must be placed in locations of B_1, and only those locations (4a)

elements from A_2 must be placed in locations of B_2 (4b)

elements placed in B_3 locations must come from A_3 (4c)

Statements (1)(2)(3) make up a QAP, and 4abc tell us that

$$x_{ij} = 0 \text{ for } i \text{ and } j \text{ such that } \begin{cases} a_i \in A_1 \text{ and } b_j \in B \setminus B_1 \\ \text{or } a_i \in A \setminus A_1 \text{ and } b_j \in B_1 \end{cases} \tag{4a}$$

$$a_i \in A_2 \text{ and } b_j \in B \setminus B_2 \tag{4b}$$

$$a_i \in A \setminus A_3 \text{ and } b_j \in B_3 \tag{4c}$$

(Here $A \setminus A_1$ is the complement of A_1 in A, etc.)

Interpretations of these constraints might be: plants of set A_2 require a water supply which is available only in locations of set B_2; locations of B_3 are small and must be occupied by plants not requiring much space, those of A_3; and plants of A_1 must be built together in a certain region B_1.

1.22 Building Layout

The building layout problem is quite similar to the facilities location problem except that we are concerned with what goes on between various rooms in a building rather than shipments between plants. One objective may be to minimize the time employees spend walking between offices or rooms by finding optimal locations for the offices. For example, in the case of a hospital, we have a wide collection of employees such as surgeons, nurses, student nurses, etc., who collect various salaries and find it necessary (according to their job) to travel with varying degrees of frequency between different rooms such as an emergency room, operating room, scrub-up room, or sterilizing room. Now it has been pointed out that, on the average, people in a typical hospital spend 38 percent of their time walking between rooms, and taking relative salaries into account, this represents 34 percent of the cost for staff salaries. If we assume that careful planning can reduce this figure by one quarter, a savings equivalent to 8½ percent of the cost for staff salaries is realized. This may amount to a substantial sum of money, especially over a period of many years [Whitehead and Elders (1964)].

There are, however, some special problems in building layout which do not occur in any analogous manner in facilities location, control panel layout, or the wiring problem. We begin in a typical building layout problem by making a list of the rooms, and the frequency with which employees of different salaries travel between the rooms. For each pair of rooms a_i and a_j we may then compute a cost factor u_{ij} equal to the sum of the products of the number of people traveling between the two rooms and their salaries (say, in dollars/hours). The movements are then examined to see if they can be eliminated as unnecessary or new facilities or organization could lead to their elimination. For example, a recovery ward may be added so the surgeon or anaesthetist doesn't need to visit patients in their own room right after an operation. A set of location units b_k is determined. Since the rooms may vary in size, some may require only one location while others require several (adjacent locations. Instead of distance $d_{k\ell}$

between locations b_k and b_ℓ, we let $d_{k\ell}$ represent the time (say, in fractions of hours) it takes to travel from b_k to b_ℓ. Of course, a difficulty arises when a room occupies several locations, and in this case we may arbitrarily take the shortest (or longest) distance between the locations used by the rooms.

Once a placement has been determined, refinements may be needed such as the addition of a corridor to help maintain aseptic conditions in the operating room, or even changes for aesthetic purposes. It should be pointed out then, that the solution of the building layout problem as a QAP (or GPP) yields only an approximate solution to the real problem. Some factors such as the cost of air ducts, plumbing, and wiring between rooms we have neglected, because in most instances the differences are small for various layouts.

1.23 Control Panel Layout

One of the objectives in designing an instrument or control panel may be to minimize eye travel distance. Assume we are assigning n instruments a_i to n locations b_k. Using notation similar to before, we let u_{ij} be the frequency of transitions from instruments a_i to a_j and $d_{k\ell}$ the Euclidean distance between locations b_k and b_ℓ. As before, the cost and assignment matrices C and H may be computed to obtain a QAP:

$$\begin{aligned} &\text{minimize} && x'Cx \\ &\text{subject to} && Hx = e_{2n} \\ & && x \text{ binary} \end{aligned}$$

This problem is discussed in [Dorris (1971), 18-21], while a linear case is given in [Freund and Sadosky (1967)].

1.24 The Wiring Problem

Beginning in the late 1950's, a problem associated with the construction of computers, now known as the backboard wiring problem, began to receive attention [Loberman and Weiberger (1957)], [Steinberg (1961)]. The problem is to find a placement (assignment) for a number of electrical components, which must be interconnected, on a board so that the total length of wire used to make the connections is minimal. An assumption is made that minimal total wire length is the best approximation for satisfying a number of goals for good circuit design, which include:

reducing capacitance
reducing delay line effects
distribution of heat sinks
avoidance of excessive wire build-up on routing channels
avoiding cross-talk without excessive use of expensive shielding
neatness to reduce construction and maintenance costs
using smallest amount of wire

We suppose a set of n electrical components a_i are given which must be assigned to a set of n locations b_k. As before, without loss of generality, it is assumed the number of components equals the number of locations, and they will be assigned in a one-to-one manner. Let u_{ij} be the number of wires connecting components a_i and a_j. The distance $d_{k\ell}$ between b_k and b_ℓ will, of course, depend on the wiring channels used and may be either Euclidean or "street" distance. The cost and assignment matrices may be found, as before, to give the problem:

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_{2n}$$

$$x \text{ binary}$$

Example: The best known example, and, in fact, about the only real problem which appears to have been studied in detail by many authors, is the one given in [Steinberg (1961)], [Graves and Whinston (1970)]. It is pointed out in [Hanan and Kurtzberg (1972)] that there is a need for studying additional realistic problems. Here we give a small example just to illustrate the ideas involved, the structure of the matrices, and some possible extensions. Consider the problem of placing four components a_i $(i = 1,2,3,4)$, with u_{ij} wire connections between a_i and a_j, in four locations b_k $(k = 1,2,3,4)$. The values of u_{ij} are given in Table 1 and the distances $d_{k\ell}$ between locations b_k and b_ℓ are given in Table 2. Figure 1 shows the actual arrangement of locations on the board. We have used Euclidean distances to compute the $d_{k\ell}$.

i \ j	1	2	3	4
1	0	3	7	1
2	3	0	2	5
3	7	2	0	3
4	1	5	3	0

Table 1. Values of u_{ij}

k \ l	1	2	3	4
1	0	1	1	$\sqrt{2}$
2	1	0	$\sqrt{2}$	1
3	1	$\sqrt{2}$	0	1
4	$\sqrt{2}$	1	1	0

Table 2. Values of $d_{k\ell}$ with Euclidean Distance

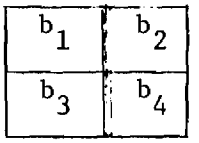


Figure 1

The assignment matrix H is

$$H = \begin{bmatrix}
1 & 1 & 1 & 1 & & & & & & & & & & & & \\
 & & & & 1 & 1 & 1 & 1 & & & & & & & & \\
 & & & & & & & & 1 & 1 & 1 & 1 & & & & \\
 & & & & & & & & & & & & 1 & 1 & 1 & 1 \\
1 & & & & 1 & & & & 1 & & & & 1 & & & \\
 & 1 & & & & 1 & & & & 1 & & & & 1 & & \\
 & & 1 & & & & 1 & & & & 1 & & & & 1 & \\
 & & & 1 & & & & 1 & & & & 1 & & & & 1
\end{bmatrix}$$

The cost matrix C could be computed as before, but since the (u_{ij}) and $(d_{k\ell})$ matrices are symmetric, we may write C as the partitioned matrix

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

where $C_{ij} = \begin{pmatrix} C^{ij}_{rs} \end{pmatrix}$ $\quad r = 1,2,3,4,$ $\quad s = 1,2,3,4$

and $C^{ij}_{rs} = \begin{cases} u_{ij} d_{rs} & \text{if } i < j \quad (i = 1,2,3 \quad j = 2,3,4) \\ & \text{and } r \neq s \\ 0 \ , & \text{otherwise} \end{cases}$

So C is the 16 x 16 matrix whose non-zero elements are shown:

$$C = \left[\begin{array}{cccc|cccc|cccc}
0 & 3 & 3 & 3\sqrt{2} & 0 & 7 & 7 & 7\sqrt{2} & 0 & 1 & 1 & \sqrt{2} \\
3 & 0 & 3\sqrt{2} & 3 & 7 & 0 & 7\sqrt{2} & 7 & 1 & 0 & \sqrt{2} & 1 \\
3 & 3\sqrt{2} & 0 & 3 & 7 & 7\sqrt{2} & 0 & 7 & 1 & \sqrt{2} & 0 & 1 \\
3\sqrt{2} & 3 & 3 & 0 & 7\sqrt{2} & 7 & 7 & 0 & \sqrt{2} & 1 & 1 & 0 \\
\hline
 & & & & 0 & 2 & 2 & 2\sqrt{2} & 0 & 5 & 5 & 5\sqrt{2} \\
 & & & & 2 & 0 & 2\sqrt{2} & 2 & 5 & 0 & 5\sqrt{2} & 5 \\
 & & & & 2 & 2\sqrt{2} & 0 & 2 & 5 & 5\sqrt{2} & 0 & 5 \\
 & & & & 2\sqrt{2} & 2 & 2 & 0 & 5\sqrt{2} & 5 & 5 & 0 \\
\hline
 & & & & & & & & 0 & 3 & 3 & 3\sqrt{2} \\
 & & & & & & & & 3 & 0 & 3\sqrt{2} & 3 \\
 & & & & & & & & 3 & 3\sqrt{2} & 0 & 3 \\
 & & & & & & & & 3\sqrt{2} & 3 & 3 & 0
\end{array}\right]$$

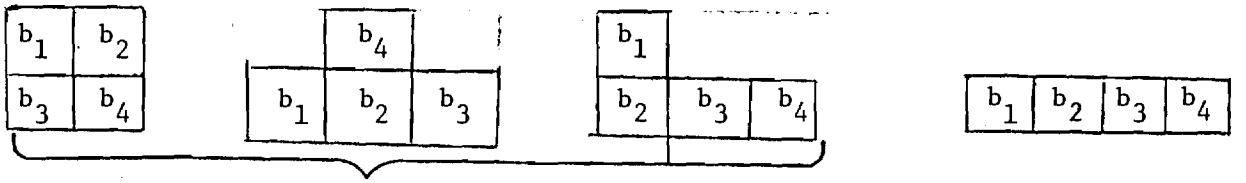
The problem is

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_8$$

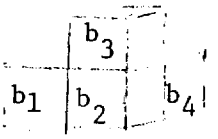
$$x \text{ binary}$$

Some Extensions The above problem may be altered by introducing weights to the u_{ij}'s, as was done in the hospital example of the building layout problem, or by changing the $(d_{k\ell})$ matrix. We may wish to change the $(d_{k\ell})$ matrix to correspond to such physical arrangements of locations as



2 dimensional arrangements 1 dimensional

or possibly a 3 dimensional layout such as



If $\mathcal{D}$ is a set of matrices

$$\mathcal{D} = \{ D = (d_{k\ell}) \}$$

we may wish to consider the problem of finding the best arrangement of locations as well:

$$\begin{array}{lll} \text{minimize} & \text{minimize} & x'C_D x \\ D \in \mathcal{D} & Hx = e & \\ & x \text{ binary} & \end{array}$$

(Here C_D indicates the cost matrix depends on D).

As was done in the facilities location problem, we may use the GPP to require certain sets of components be located in specified sets of locations. This may be useful to keep the components of one section, say an amplification or power section, of a piece of electrical equipment together, as an aid to construction or maintenance.

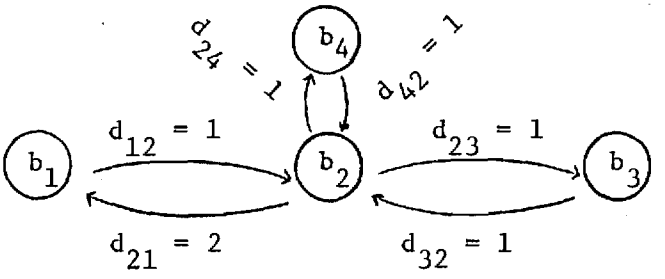
1.3 The Traveling Salesman Problem

Suppose a salesman is given a list of n cities connected by a number of roads of known distance. His problem, known as the traveling salesman problem, is to find a route of minimal length he can follow so as to visit each city once and only once, and return to where he started. (Hu (1969), 270-271], [Garfinkel and Neurhauser (1972), 354-361].

1.31 The Classical Traveling Salesman Problem

There are several different versions and formulations of the TSP, and the one described above appears to be the original. We are concerned with stating the problem as a QAP, but first mention, for clarification, some of the variations. First, we are considering what is called the closed tour version. This simply means that the salesman must end up where he started. An alternative, is the open tour version in which the salesman starts in a given city, visits each city once and only once, but does not return to the original city. It is easy to convert the closed tour version to the open tour, by dividing the starting city into two cities. We are just going to consider the closed tour here.

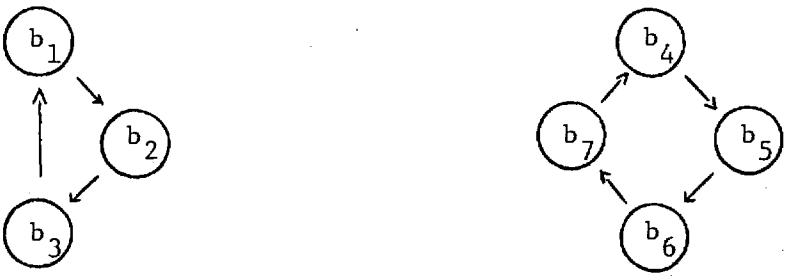
Now the following example is a simple TSP which has no solution. We are given four cities b_k (k = 1,2,3,4) and some roads of known distances $d_{k\ell}$ between them as shown below



It is easy to see that it is impossible to visit all four cities and not visit b_2 three times. There is a modification of the TSP described in 1.32 which allows for the possiblity of passing through cities, but as the classical TSP, this problem has no solutions. We also note that $d_{12} = 1$ and $d_{21} = 2$, so that $d_{12} \neq d_{21}$. It is not necessary that the distances between two cities be the same each way. The idea of a subtour comes from formulations of the TSP in which variables x_{ij} are used:

$$x_{ij} = \begin{cases} 1 & \text{if the salesman travels from } b_i \text{ to } b_j \\ 0 & \text{otherwise} \end{cases}$$

Unless constraints are added to prevent it, a "solution" may occur consisting of more than one trip, as shown below:



The trips 1-2-3-1 and 4-5-6-7-4 are called subtours, and obviously cannot be a solution to any problem in which a salesman must begin and end his trip in the same city. A tour is a sequence for visiting each city exactly once and ending up in the starting city. In the formulation we now give for the TSP as a QAP, the problem of subtours does not arise.

Suppose we are given n cities b_k $(k = 1,2,\ldots,n)$ with distances $d_{k\ell}$ from city b_k to b_ℓ. We assume $d_{k\ell} \geq 0$, and in cases where there is no $d_{k\ell}$ given (as when there is no direct road from b_k to b_ℓ) we assign some large number M. We think of the TSP as a facilites location problem in which n plants a_i $(i = 1,2,\ldots,n)$ are to be assigned to the n cities b_k $(k = 1,2,\ldots,n)$ under the conditions the units shipped from a_i to a_j, u_{ij} are given by:

i)	for $i = 1,2,\ldots,n-1$	$u_{i\ i+1} = 1$
ii)		$u_{n\ 1} = 1$
iii)	all others	$u_{ij} = 0$

and we wish to minimize the total cost. Constructing cost and assignment matrices C and H as before, we have the TSP stated as the QAP

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_{2n}$$

$$x \text{ binary}$$

A variable $x^*_{ik} = 1$ in the final solution x^* means city k is visited as the ith city.

1.32 Four Modifications of the Traveling Salesman Problem

As noted in 1.31 there are some TSP's which have no solution because it is impossible to visit each city exactly once. There are other examples in which we could reduce the distance traveled if we could "pass through" a city, i.e., visit it more than once. Such an example is shown below in Figure 1.

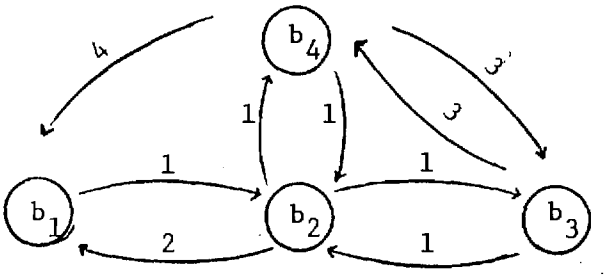
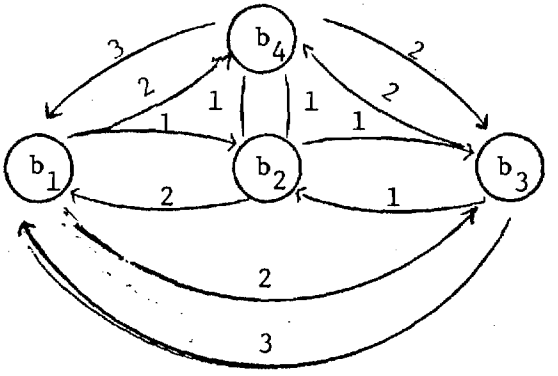


Figure 1

The shortest route to visit all cities by the TSP is 1-2-3-4-1 and has length 9. If we now permitted to visit b_2 three times, we could choose the route 1-2-4-2-3-2-1 which has length 7. One method to avoid having the larger soltuion to the TSP forced upon us, which would be desirable in most real applications, is to construct an expanded TSP as follows. Given n cities and some values $d_{k\ell}$ representing the distances of roads from city b_k to b_ℓ, define values of $d_{k\ell}$ for all $k = 1,2,\ldots,n$; $\ell = 1,2,\ldots,n$ by

$$\tilde{d}_{k\ell} = \begin{cases} \text{the shortest distance from city } b_k \text{ to } b_\ell & \text{if } k \neq \ell \\ \infty & \text{if } k = \ell \end{cases}$$

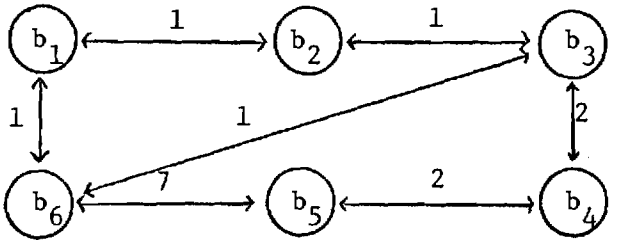
(We will need to keep track of the shortest paths when stating the solution of the original problem from the expanded problem). The example in Figure 1 would then become



with the optimal solution 1-2-4-3-1 and distance 7. If the original problem in Figure 1 had satisfied the triangle inequality for distances*

$$d_{ij} + d_{jk} \geq d_{ik}$$

and we had been given values for <u>all</u> $d_{k\ell}$ ($k = 1,2,\ldots,n$; $\ell = 1,2,\ldots n$), then the optimal routing wouldn't require visiting b_2 three times and the expanded TSP wouldn't have been needed [Hu (1969), 270]. The following example is one in which the given $d_{k\ell}$ satisfies the triangle inequality, but a city is visited twice to obtain the shortest route 1-2-3-4-5-4-3-6-1 (length 12). The reason here is that not all the inter-distances among the cities are given.



We now mention <u>some extensions of the TSP</u>. In addition to being given n cities b_k ($k = 1,2,\ldots,n$) and some distances $d_{k\ell}$ between cities b_k and b_ℓ, we may be given times $t_{k\ell}$ between cities b_k and b_ℓ and due dates τ_k by which a city b_k must be visited. For a given tour ψ, let $b_{\psi(i)}$ be the ith city visited, and suppose $b_{\psi(i)}$ is visited at time

$$\hat{\tau}_{\psi(i)} = t_{\psi(0)\psi(1)} + t_{\psi(1)\psi(2)} + \cdots + t_{\psi(i-1)\psi(i)}$$

for $i = 1,2,\ldots,n$. If we define the <u>tardiness</u> in visiting a city as

*It doesn't: $2 = 1 + 1 + d_{3,2} + d_{2,4} < d_{3,4} = 3$

$\hat{\tau}_k - \tau_k$, one objective may be to find a tour ψ which minimizes the maximum tardiness. Another problem may be to minimize a linear combination of the distance traveled and the maximum tardiness. The GPP may be used to make such requirements as the first city visited must be one of three selected cities, or the last two cities visited must include a certain given city.

In a fourth modification of the TSP, considered by [Srivastava (1969)], the set of cities $B = \{b_1, b_2, \ldots, b_n\}$ is divided into a home city and a family of s collectively exhaustive and mutually exclusive sets of cities: $B = \{\text{home city}\} \cup \bigcup_{i=1}^{s} B_i$. The problem is to determine an optimal tour, starting at the home city, which includes at least one city from each of the sets B_i $(i = 1,2,\ldots,s)$. An application of this modification, in which a salesman has to complete multiple jobs subject to competition, is given in [Garg, etc. (1970)].

1.33 Production Scheduling Application

Suppose we have n tasks b_k $(k = 1,2,\ldots,n)$ which are to be performed on some machine. A change over time $d_{k\ell}$ is required to chage the machine from doing task b_k to task b_ℓ. The problem of selecting a sequence in which to perform the tasks and return the machine to its original state so the sum of changeover times is minimized is a TSP [Maxwell (1964)].

An n-job, m-machine scheduling problem is modelled as a TSP in [Wismer (1972)].

1.34 The Multi-Traveling Salesmen Problem, Its Generalizations, and Some Applications

The TSP is actually a special case of the multi-traveling salesmen problem (M-TSP) [Svestka and Huckfeldt (1973)]. In the M-TSP we are given n cities b_k $(k = 1,2,\ldots,n)$, one of which is designated as a "home city", and distances $d_{k\ell}$ between cities b_k and b_ℓ. There are m salesmen instead of one, and the problem is to determine routes for each of the m salesmen to follow so that every city (except the home city) is visited exactly once by exactly one salesman, the home city is included in each salesman's route, and the total distance traveled by all salesmen is a minimum. We note that in the M-TSP a "home city" is designated beforehand. Three alternatives are to designate several home cities, as is done in the multi-terminal delivery problem [Tillman and Hering (1971)], or allow for either one or several undesignated home cities which will be determined as

part of the problem. (Not requiring a home city actually falls into the last two cases). Another generalization of the M-TSP is given in [Zak (1972)] where each salesman may have a different distance matrix.

We now give some applications of the M-TSP and its generalizations.

The Bank Messenger Scheduling Problem

In this problem a crew of messengers collects deposits at branch banks and returns them to a central office for processing. If there are m messengers and n branches, this is the M-TSP. The Cleveland Trust Company, Cleveland, Ohio, has actually made use of the M-TSP to determine routes for its messengers [Svestka and Huckfeldt (1973)].

Printing Press Scheduling for Multi-Edition Periodicals

A printing press scheduling problem is given as a 3-traveling salesmen problem with a number of side constraints in [Gorenstein (1970)]. But even a simplified version of this problem is not solvable for problem sizes with practical applications.

Scheduling Excavators Movements

The scheduling of the movements of excavators in mine pits is given in [Tsoi, etc. (1971)] as a M-TSP. Cases with route length and cycle constraints are given.

Single Terminal Delivery Problem

In this problem delivery trucks supply customers from a single terminal. This is a M-TSP, with the restriction that the trucks have limited capactiy and need to return to the home city (terminal) to refill.

Multiterminal Delivery Problem

This is an application of the generalization of the M-TSP in which several home cities are designated. While in the single terminal delivery problem we assign customers to routes so total distance is minimized, in the multiterminal delivery problem we need to assign customers to routes and routes to terminals. This problem and references for the single terminal case may be found in [Tillman and Hering (1971)].

1.4 Applications of the QSCP and Related Problems

In this section we give a number of examples of set covering problems, most of which were originally formulated as LSCP (or LSPP) but can be better formulated as QSCP (or QSPP). These include the delivery problem, location of fire hydrants, airline crew scheduling, assembly line balancing problem, information retrieval, project selection, capital budgeting, and attack/defense networks. A general structure common to all of these applications is first given, then each application is considered separately. The project selection problem is considered in detail and some directions are indicated for the formulation of a more realistic model.

We recall that the QSCP was

$$\text{minimize} \quad x'Cx \tag{1}$$

$$\text{such that} \quad Ax \geq b \tag{2}$$

$$x \text{ is a binary vector} \tag{3}$$

where x, C, A, and b were described in §1-1. A was an m x n matrix. In the applications each of the m rows is identified with either a client (delivery problem), a flight segment (airline crew scheduling), a task (assembly line balancing), or same activity. Various subsets of the clients flight segments, tasks, or activities are selected to form (respectively) routes, trips, assembly stations, or projects, which are identified with the columns of A. For example, in the delivery problem, if client i is included in route j, then $a_{ij} = 1$, if not, $a_{ij} = 0$. The b vector (m x 1) has element $b_i = 1$ in a particular problem if we wish the ith client to be included in the final solution. In the set covering problem, $b_i = 0$ doesn't necessarily preclude the possibility of client i from being taken care of, as it does in the partitioning problem, but it doesn't require that client i be taken care of. If we don't care if a client is served several times we would use a covering problem; if we want to restrict a client to one service in a problem, the partitioning problem would be used. In the LSCP and LSPP, costs are assigned to the routes, trips, assembly stations, or projects (i.e. the columns of A). The quadratic versions of these problems permit additional costs or discounts for various pairs of columns selected. In some applications there are good reasons why two activities should cost less than the sum of the costs of the activities

individually, while in other cases there appears to be no reason for extending the linear formulation to a quadratic one.

1.41 Routing Problems: Truck Delivery and School Bussing

In the truck delivery and school bussing problems we are concerned with the delivery or pick-up of items, or, in the case of school bussing, children. There are a number of delivery or pick-up stations, each of which must be visited at least once, and a central terminal or school from which the items are sent out or collected. Various subsets of the delivery stations are formed to make up routes, and each route has a cost associated with it. The problem of selecting a set of routes so that each station is included in at least one route is a LSCP, or, if reasons can be found for quadratic interaction of the costs, a QSCP. In the case of school bussing one reason to use a QSCP might be to help achieve a racial balance. The question remains of how to do this. This would allow certain combinations of routes to be desirable and others to be undesirable. Since trucks or buses have limited capacity, the number of items picked up or delivered on a route is limited; also, since the total number of available trucks is limited (not allowing multiple trips for a vehicle), we may need a solution in which the total number of routes selected is less than or equal to the total number of vehicles [Balinski and Quandt (1964)], [Murty (1973)].

1.42 Location of Fire Hydrants

The problem of locating fire hydrants is discussed in [Murty (1973)] We consider a network whose arcs are streets, and nodes are intersections of the streets. The problem is to select a subset of the nodes for placement of fire hydrants so each street has at least one fire hydrant. This problem is probably best left as a LSCP.

1.43 Airline Crew Scheduling

In the airline crew scheduling problem flight segments are selected to form trips, and the idea is to select trips to cover all flight segments. This is a LSCP. See [Murty (1973)] and [Arabeyre, etc. (1969)].

1.44 Assembly line balancing

Tasks are combined to be performed at assembly stations in the assembly line balancing problem. The problem is to determine which assembly stations are to be set up. This is a partitioning problem, since a task is to be

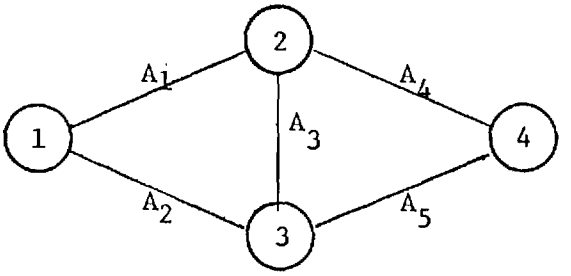
performed exactly once. Complications arise in the form of precedence relations on tasks and due dates by which tasks must be performed. [Murty (1973)], [Salvesan (1955)].

1.45 Information Retrieval

The information retrieval problem is how to select a number of tapes from a collection of tapes, each of which contains subsets of required information, so that all desired information may be found searching a minimal length of tape. This is a LSCP and there appears to be no advantages in considering it as a QSCP. See [Day (1965)].

1.46 Attack/Defense Networks

Attack and defense of networks may be formulated as QSCP. Consider the network shown below in Figure 1.



The nodes in the network are thought of as strategic points; while the arcs represent communication lines such as roads, bridges, pipelines, etc. The objective is to attack the network in a optimal manner so that nodes 1 and 4 are separated. If A_i represents arc i and x_i the associated variable, then $x_i = 1$ means A_i is destroyed and $x_i = 0$ means A_i is not destroyed. The constraint matrix which requires every path from node 1 to node 4 be cut at least once is

$$\begin{array}{c} \begin{matrix} A_1 & A_2 & A_3 & A_4 & A_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

If a cost is assigned to destroying each arc, a linear objective function may be obtained to give a LSCP. However, the cost of destroying two arcs, say A_1 and A_5, may be greater than the sum of the costs of destroying them individually. For example, they may be far apart and require some scarce resource to destroy. In this case a quadratic objective function $x'Cx$ is used to consider such interactions. The element c_{15} would be positive and represent the additional cost of cutting A_1 and A_5. The attack problem is a good example of a LSCP which is better modelled as a QSCP. The defense problem, how to optimally defend a network which will be attacked, by the construction of more arcs subject cut limitations, is also a QSCP. [Bellmore and Ratliff (1971a), (1971b)], [Jarvis (1968)].

1.47 Project Selection and Capital Budgeting

We consider the problem where a number of projects are combined to form activities. A set of activities is to be selected in an optimal manner so that all required projects are included among the selected activities. Since it would probably be desired that a project is not included more than once, a partitioning problem would be used. Since there may be advantages of selecting certain pairs of activities, a QSCP is used.

To give some insight into the discount cost arising from the selection of two activities, consider the interaction which occurs in construction activities. There are three primary reasons why two activities are sometimes cheaper than the sum of the individual costs:

1. a materials discount When two activities contain projects which use the same type of material, a savings may be realized by buying this material in bulk. There is a limit to this savings, e.g. three \$5,000 projects may lead to a savings, but the addition of two more \$5,000 projects may not be significant.
2. already having equipment on the construction site To move a piece of machinery may cost \$400, \$100 to get a truck on the site and \$300 to move the machine. Perhaps three machines could be moved at the same time for 100 + 3(300) = \$1000 or a \$200 savings over 3(400) = \$1200.
3. insurance savings (four jobs cost only a little more than one).

Unfortunately, there are also three major reasons why the QSPP doesn't

apply in many cases. First, we assume we know all of the projects (and activities) from the start, while in real problems new projects arise as work precedes on old ones. Then we assume all the projects are funded ahead of time and from one source. In practice, funding comes from several different sources, cannot be mixed, and is in various stages of uncertainty. Finally, as in construction work, there is a scheduling problem. We may need to hire two or three contractors at once, because one may not be able to simultaneously work on three projects which must be done by a certain date.

In practice then, a person with, say three, projects would probably

1. do none of them
2. do them all together, if possible
3. do separately as required by funding and scheduling.

Additional complications may occur in the project selection/capital budgeting problem in that

1. the selection of one activity may either necessitate or preclude the possiblity of the selection of another activity
2. the total number of activities is limited
3. the total amount of money is limited (perhaps cancelling some optional activities).

It appears, in view of the difficulties described above, that a new model which takes the scheduling and funding, at least, into account may be developed. [Alexander (1973)] [Weingartner (1966)].

1.48 Political Districting

In political districting an area (e.g. a state) is partitioned into a number of smaller areas called districts which are assigned a number of representatives. The area has also been partitioned into a number of basic population units (e.g. counties). A political districting plan consists in assigning the population units to the districts so that:

i) a population unit is assigned to exactly one district
ii) the total population assigned to a district doesn't deviate from the mean district population by more than a given number
iii) a district is contiguous, i.e. it is possible to walk

between population units making up the district without going into another district (roughly speaking)

and iv) a district is compact, i.e. somewhat circular or square

Details and examples are given in [Garfinkel and Neurhauser (1970)]. Without the contiguity and compactness restrictions, this problem may be formulated as a QSCP. Suppose we have n basic population units a_i (i = 1,2,...,n) with the population of unit a_i being d_i, and m districts b_k (k = 1,2,...,m). Define variables x_{ik} by

$$x_{ik} = \begin{cases} 1 & \text{if } a_i \text{ is assigned to } b_k \\ 0 & \text{otherwise} \end{cases}$$

and let $x' = (x_{11}\ x_{12} \cdots x_{1m}\ ;\ x_{21}\ x_{22} \cdots x_{2m}\ ;\ \cdots\ ;\ x_{n1}\ x_{n2} \cdots x_{nm})$

The QSCP is then:

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_n$$

$$\text{and } x \text{ a binary vector}$$

where C is the $(n \cdot m) \times (n \cdot m)$ partitioned matrix

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \cdots & C_{2m} \\ \vdots & & & \\ C_{m1} & & & C_{mm} \end{bmatrix}$$

$$C_{ij} = \begin{bmatrix} d_1^2-2dd_1 & d_1d_2 & \cdots & d_1d_n \\ d_1d_2 & d_2^2-2dd_2 & & d_2d_n \\ \vdots & \vdots & \ddots & \\ d_nd_1 & d_nd_2 & \cdots & d_n^2-2dd_n \end{bmatrix} \quad \text{if } i = j$$

and C_{ij} = the n x n zero matrix for $i \neq j$

(here $d = \frac{1}{m} \sum_{i=1}^{n} d_i$, the mean district population)

and H is the n x (n·m) matrix

$$H = \begin{bmatrix} e_m' & & & \\ & e_m' & & \\ & & \ddots & \\ & & & e_m' \end{bmatrix} .$$

1.5 Miscellaneous Applications

The coloring problem [Bessiere (1965)], determining the minimum number of colors needed to color a map so that no two adjacent areas have the same color, is a LSPP. [Lawler (1960)] discusses the problem of minimizing "latency" in magnetic drum on disc storage computers as a QAP. The reader may see applications to staff and class scheduling, constrained least squares estimation, or message switching centers.

1.6 Some Comments on the Practicability of QSCP Applications

In this section we discuss the realistic values of the applications given in sections 1.2, 1.3, and 1.4. Quite often, simple examples are used to illustrate the application of some theory, and, while they serve the purpose of clarifying the theory, they are of no practical use. Basically there are three reasons for this:

i) the real problem is too simple to waste time applying a complicated theory

ii) the theory is not complicated enough to model the real problem

iii) the theory is OK for the real problem but leads to computationally infeasible problems.

We expect the first case (i) to hold for many facilities location and building layout problems. For example, a motel is a relatively uncomplicated structure and probably wouldn't require a QAP to determine a layout. The true value of the QAP is apparent when a complicated structure such as a hospital is designed or a complex wiring problem is considered. Unfortunately, in these cases the problem may become computationally infeasible (iii). One of the purposes of this study is to deal with such problems.

A good example where (ii) the theory was not complicated enough is the project selection problem of section 1.47. Directions were indicated here for a new problem.

Many of the references cited state that computationally infeasible problems result from the application of the theory described to real problems: [Jarvis (1968),88] states that although the attacker problem can be handled in a reasonably efficient manner, the defender problem cannot. Similar statements occur in [Maxwell (1964)] concerning production scheduling, in [Gorenstein (1970)] on printing press scheduling, and in [Freund and Sadosky (1967)] on control panel layout. In other words, there are a large number of valuable applications, once efficient solution procedures are developed. These procedures may include new algorithms and combinations/modifications of old ones, or the attempt to include more complications of the original problem as in the GPP.

How about the use of quadratic verse linear models for some applications? As was mentioned in section 1.4 some problems, e.g. the information retrieval problem, are naturally linear and no basis was found to consider quadratic

interaction. Examples of applications of set covering problems with quadratic objective functions, with the exception of the special case of the QAP, are not easy to find. Most examples considered, which were not linear, such as construction project selection, exhibit higher order interactions. It does not appear that examples with high order interactions can be approximated by quadratic interactions in a successful manner. Although [Weingartner (1966)] points out the tremenous advantage of the quadratic over the linear model for project selection/capital budgeting, some concrete, real applications remain to be found which can be solved as, say, a QSPP.

REFERENCES

Alexander, D., Interview, Physical Plant, Georgia Institute of Technology, 1973.

Arabeyre, J. P., J. Fearnley, F. C. Steiger, and W. Teather, "The Airline Crew Scheduling Problem: A Survey", Transportation Science 3, 140-163, 1969.

Balinski, M. L. and R. E. Quandt, "On an Integer Program for a Delivery Problem", Operations Research 13, 300-304, 1964.

Bellmore, M. and H. D. Ratliff, "Set Covering and Involutory Bases", Man. Sci. 18, 194-205, 1971a.

Bellmore, M. and H. D. Ratliff, "Optimal Defense of Multi-Commodity Networks", Man. Sci. 18, 174-185, 1971b.

Bessiere, F., "Sur la Recherche der Nombre Chromatique d'Un Graph par uu Programme Lineaire en Nombres Entiers", Rev. Franc, Recherche Operat. 9, 143-148, 1965.

Day, K. H., "On Optimal Extracting from a Multiple File Data Storage System: An Application of Integer Programming," Operations Research 13, 482-495, 1965.

Dorris, A. L., "The Utility of Optimization Techniques in the Design of Man-Machine Systems", Master's Thesis, Georgia Institute of Technology, 1971.

Freund, L. E. and T. L. Sadosky, "Linear Programming Applied to Optimization of Instrument Panel and Workplace Layout", Human Factors 9, 295-300, 1967.

Garfinkel, K. S. and G. L. Nemhauser, "Optimal Political Districting by Implicit Enumeration Techniques", Man. Sci., 16, 495-504, 1970b.

Garfinkel, K. S. and G. L. Nemhauser, Integer Programming, John Wiley & Son, 1972.

Garg, K. C., S. Kumar, P. Dass, and P. Len, "Generalized Traveling Salesman Problem Through n Sets of Nodes in a Competitive Market", Ablanf und Plassungsforschung (Germany) 11, 116-120, 1970.

Geddes, P. H., "The Solution of Large Size Quadratic Set Covering Problems", Master's Thesis (in progress), Georgia Institute of Technology, 1974.

Gorenstein, S., "Printing Press Scheduling for Multi-Edition Periodicals", Man. Sci. 16, 373-383, 1970b.

Graves, G. W., and A. B. Whinston, "An Algorithm for the Quadratic Assignment Problem" in Integer and Nonlinear Programming, ed. by J. Abadie, North Holland Pub. Co., Amsterdam, 473-497, 1970.

Hanan, M., and J. M. Kurtzberg (1972), "A Review of the Placement and Quadratic Assignment Problems", SIAM Review 14, 324-342, 1972.

Hu, T. C., Integer Programming and Network Flows, Addison-Wesley, 1969.

Jarvis, J. J., Optimal Attack and Defense of a Command and Control Communications Network, Ph.D. Dissertation, The Johns Hopkins University, 1968.

Lawler, E. D., "Notes on the Quadratic Assignment Problem", Harvard Computation Laboratory, unpublished paper, 1960.

Loberman, H., and A. Weinberger, "Formal Procedures for Connecting Terminals with a Minimum Total Wire Length", Journal of the Association of Computing Machinery 4, 428-437, 1957.

Maxwell, W. L., "The Scheduling of Single Machine Systems: A Review", The International Journal of Production Research 3, 177-199, 1964

Murty, K. G., "On the Set Representation and Set Covering Problems", in Symposium on the Theory of Scheduling and its Applications, ed. by M. Beckman, G. Goos, and H. P. Kunzi, Springer-Verlag, 143-163, 1973.

Salvesan, M. E., "The Assembly Line Balancing Problem", Journal of Industrial Engineering 6, 18-25, 1955.

Srivastava, S. S., S. Kumar, K. C. Garg, and P. Len, "Generalized Traveling Salesman Problem through n Sets of Nodes", CORS Journal 7, 97-101, 1969.

Steinberg, L., "The Backboard Wiring Problem: A Placement Algorithm", SIAM Review 3, 37-50, 1961.

Svestka, J. A. and V. E. Huckfeld, "Computational Experience with an M-Salesman Traveling Salesman Algorithm", Man. Sci. 19, 790-799, 1973.

Tillman, F. A., and R. W. Hering, "A Study of a Look-Ahead Procedure for Solving the Multiterminal Delivery Problem", Transportation Research 5, 225-229, 1971.

Tsoi, A. KH., S. M. Taskai, and V. N. Druzhinin, "The Scheduling of Movements of Excavators as a Problem of Several Traveling Salesmen", Trudy Instituta Gormogo Dela, Akademirijo Nauk Kazakhrkogo SSR (U.S.S.R.) 45, 9-13, 1971.

Weingartner, H. M., "Capital Budgeting of Interrelated Projects: Survey and Synthesis", Man Sci. 12, 458-516, 1966.

Whitehead, N. and M. Z. Elders, " An Approach to the Optimum Layout of Single-Storey Buildings", Architects' Journal 139, 1373-1380, 1964.

Wismer, D. A., "Solution of the Flowshop-Scheduling Problem with No Intermediate Queues", Operations Research 20, 689-697, 1972.

Zak, Yu. A., Solution Algorithms for "n Traveling Salesmen" Problem, Kibernetika (U.S.S.R.) 1, 99-106, (Russian, English Summary), 1972.

Section II

QUADRATIC ASSIGNMENT ALGORITHM

In this section, we will describe a branch and bound algorithm for solving the quadratic assignment problem. Recall that the problem is to assign m objects to n locations ($n \geq m$). If object i is assigned to location $\ell(i)$, a fixed cost $f_{i\ell(i)}$ is encountered. A flow or interaction u_{ij} between objects i and j results in a cost of the form $u_{ij}d_{\ell(i)\ell(j)}$, where $d_{\ell(i)\ell(j)}$ is the distance between locations $\ell(i)$ and $\ell(j)$. The total cost to be minimized can then be stated as follows.

$$\text{Minimize} \quad \sum_{i=1}^{m} f_{i\ell(i)} + \sum_{j=1}^{m} \sum_{\substack{i=1 \\ j\neq i}}^{m} u_{ij}d_{\ell(i)\ell(j)}$$

Decision Tree and General Framework

Since the problem is combinatorial in nature, all possible ways of assigning m objects to n locations, must be considered. Doing this explicitly, however, will result in a huge number of possibilities, and is impossible from a computational point of view. A large number of these possibilities must not be enumerated explicitly, if one is to develop a decent procedure. At each stage of the algorithm, we have a set of objects that has already been assigned to certain locations. These already assigned objects form a partial solution of the assignment problem. In order to obtain a feasible solution, i.e. a complete assignment, we must find a completion of the partial solution. Rather than considering all the possible ways of completing the partial solution, we first investigate whether the partial solution may lead to a complete solution, with an objective value which is smaller than the best complete solution that we already have. This is done by calculating a lower bound on the cost of the partial solution plus the cost of completing this partial solution.

Let B be the lower bound and let C* be the cost of the best available assignment. If $B \geq C^*$ then any completion of the partial solution can lead to no improvement. In this case, the partial solution is said to be fathomed, and is abandoned. On the other hand, if $B < C^*$, it is worthwhile to pursue the partial solution, by seeking to assign more objects.

Calculation of Lower Bounds On Cost of Completing a Partial Solution

Suppose that a set of objects in the set I have already been assigned to a set of locations J. In particular, suppose that object i is assigned to location $\ell_{(i)}$. The cost of this partial solution, C_1, can be determined exactly as follows.

$$C_1 = \sum_{i \in I} f_{i\ell(i)} + \sum_{\substack{j \in I \\ j \neq i}} \sum_{i \in I} u_{ij} d_{\ell(i)\ell(j)}$$

Note that C_1 consists of the fixed cost of assigning objects in the set I to their locations, plus the interaction among these objects.

The cost of completing this partial solution, i.e. the addtional cost resulting from assigning the remaining objects, consist of the fixed costs, plus two types of interaction costs. The interaction from unassigned objects to assigned objects, with bound C_2, plus the interaction between the unassigned objects themselves, with bound C_3.

C_2: bound on cost from unassigned objects to assigned objects. C_2 is the optimal cost of the following linear assignment problem.

$$\text{Minimize} \quad \sum_{j \notin J} \sum_{i \notin I} b_{ij} x_{ij}$$

$$\text{Subject to} \quad \sum_{j \notin J} x_{ij} = 1 \qquad \text{for each } i \notin I$$

$$\sum_{i \notin I} x_{ij} \leq 1 \qquad \text{for each } j \notin J$$

$$x_{ij} \geq 0 \qquad \text{for each } i \notin I,\ j \notin J$$

where b_{ij} is a bound on the cost resulting from assigning the unassigned object i to the unassigned location j.

e.g., $b_{ij} = f_{ij} + \sum_{t \in J} u_{it} d_{j\ell(t)} + \sum_{t \in J} u_{ti} d_{\ell(t)j}$

In words, b_{ij} is the fixed cost of assigning object i to location j, plus the interaction from the assigned objects and object i, weighted by the distance from location j to the locations of the already assigned objects.

Of course, a complete solution of the linear assignment problem can be replaced by the simpler problem of reducing the matrix (b_{ij}) such that it has a zero in each row and each column by subtracting the minima of the rows from the rows, and the minima of the columns from the resulting columns. C_2 can then be replaced by the sum of the row minima plus the sum of the smallest $m - |I|$ column minima, where $|I|$ is the number of already assigned objects. Of course, this calculation is simpler, but the bound is not as tight as the bound obtained from solving the assignment problem.

C_3: bound on cost between unassigned objects and themselves.

In order to calculate C_3, rank the interactions u_{ij} in a descending order for $i,j \notin I$, and rank the distances d_{ij} in an ascending order for $i,j \notin J$. This results in two vectors of size $(m - |I|)^2$ and $(n - |J|)^2$. If $n > m$, complete the size of the first vector such that it has $(n - |J|)^2$ entries by adding zeros to it from the bottom. Then C_3 is the inner product of the two vectors. In words, C_3 is calculated by matching the largest interaction among unassigned elements to the smallest distance between unassigned locations, and 2nd largest interaction to 2nd smallest distance, and so forth. Clearly, this procedure will give a lower bound on the cost among unassigned elements and themselves.

Now we can calculate the overall bound B as the cost of the partial solution plus all possible completions, namely, $B = C_1 + C_2 + C_3$.

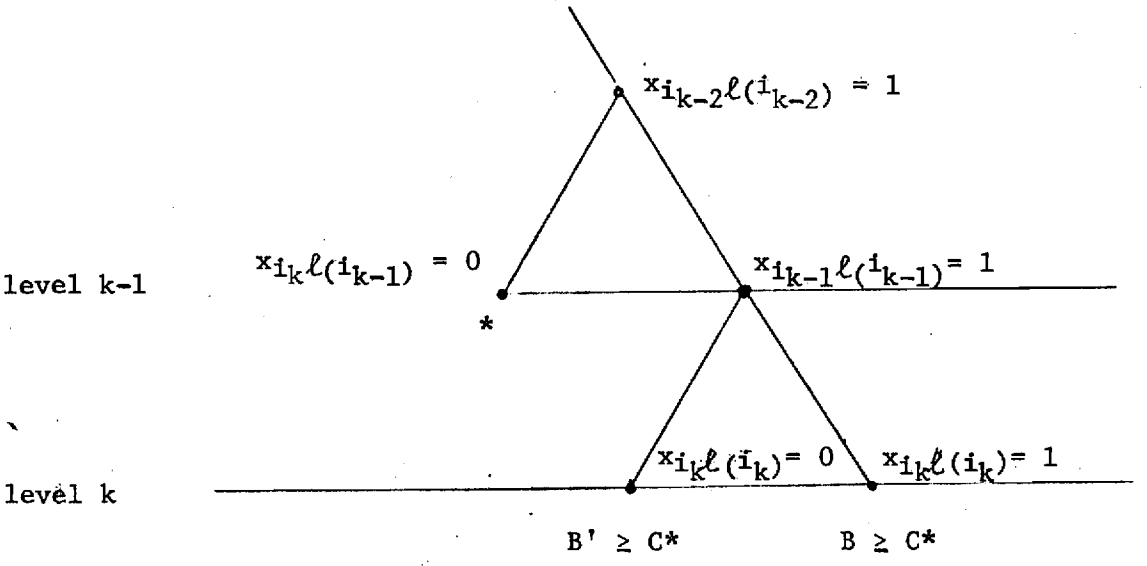
Continuation of the Search

A. Fathoming (Backward Move)

Suppose that k objects indexed by the set I have already been assigned to k locations indexed by the set J. The level of the tree is called k. A bound on the cost resulting from all completions of the current partial solution B is calculated as discussed earlier. If $B \geq C^*$, where C^* is the current best known cost of a complete assignment, then pursuing this partial solution is not worthwhile. The partial solution cannot lead to an improved complete solution, and is said to be fathomed. The last assignment, i.e. the kth assignment is banned, or prohibited, in the hope that this will lead to an improved completion. For example, if the kth assignment involved placing object i_k in location $\ell_{(i_k)}$, then $x_{i_k \ell(i_k)}$ is forced to be zero. i_k is then placed in the unassigned list of objects, (i.e. i_k removed from I), and similarly $\ell_{(i_k)}$ is removed from the unassigned list of locations (i.e. $\ell_{(i_k)}$ is removed from J). A new bound B' is calculated, in exactly the same manner as explained above, except of course, the assignment $x_{i_k \ell(i_k)} = 1$ is prohibited, say be forcing $b_{i_k \ell(i_k)} = +\infty$ while solving the linear assignment problem. If B' is still $\geq C^*$, then the partial solution of the first k-1 assignments, while banning the assignment i_k to $\ell_{(i_k)}$, can still lead to no improved solutions. Since the first k-1 assignments with $x_{i_k \ell(i_k)} = 1$ and $x_{i_k \ell(i_k)} = 0$ leads to no improvement, then all the possibilities at level k have been exhausted, and prohibiting the assignment at level $k-1$ is now possible. This is called strong fathoming, and the level of the tree is thus reduced by 1 unit, by prohibiting the assignment at level $k-1$. If i_{k-1} is assigned to $\ell_{(i_{k-1})}$, then i_{k-1} is removed from I, $\ell_{(i_{k-1})}$ is removed from J, $b_{i_{k-1} \ell(i_{k-1})} = +\infty$. Of course,

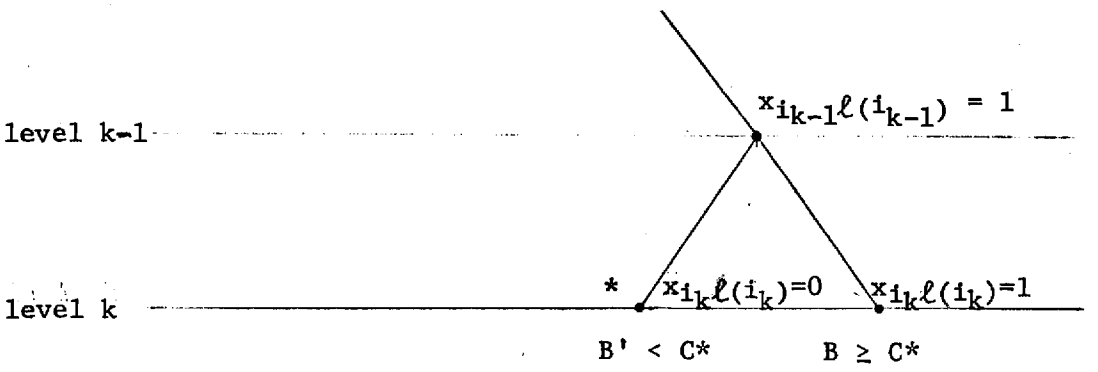
with this restriction, the assignment $x_{i_k \ell(i_k)}$ is now possible, and so $b_{i_k \ell(i_k)}$ is calculated in the usual manner. The process is then repeated. If on the other hand, the bound B' resulting from prohibiting the assignment i_k to $\ell_{(i_k)}$, is less than C*, which will be referred to as weak fathoming, then object i_k is assigned to some other unassigned location. This is discussed in more detail in the forward move of the search

The cases of strong and weak fathoming are depicted below.



Strong Fathoming

* Continue at level k-1 by trying to assign object i_{k-1} to some unassigned location $\neq \ell_{(i_{k-1})}$.



Weak Fathoming

* Continue at level k by trying to assign object i_k to some unassigned location $\neq \ell_{(i_k)}$,

B. Progress (Forward Move)

If the bound B on all completions of the partial solution at hand is less than C*, then it is not impossible to find a completion of the partial solution with an objective less than C*. At any rate, suppose that the current level of the tree is k. In other words, suppose that k objects have already been assigned, and we are trying to assign another object. Now we have a choice on which object to try to assign. Various possiblilities exist. For example, we may choose i_{k+1} to be an unassigned object with maximum interactions with already assigned objects, or choose i_{k+1} to be an unassigned object with maximum interactions with most recently assigned object i_k. At any rate, we pick a suitable unassigned object i_{k+1}. This object is assigned to an unassigned location which is not prohibited for object i_{k+1}. This location $\ell_{(i_{k+1})}$ can be picked in such a way that the total weighted interaction with assigned objects is minimal, e.g. choose $\ell_{(i_{k+1})}$ which minimizes $\sum_{j=1}^{k} u_{i_{k+1}i_j} d_{t\ell(i_j)} + \sum_{j=1}^{k} u_{i_j i_{k+1}} d_{\ell(i_j)t}$ over $t \notin J$ such that $x_{i_{k+1}t} = 1$ is not prohibited.

Choosing an object i_{k+1} and a location $\ell_{(i_{k+1})}$ in the above fashion, increases the level of the tree by 1. A bound on completing the partial solution is calculated as before and the process is repeated.

Needless to say, when the level of the tree is m, and if the cost is less than C*, then C* is replaced by the cost, and the corresponding assignment is stored.

Termination

We have described forward and backward progress of the tree search. If the level of the tree ever reaches value zero, then we stop. This would

mean that we are currently at level one, and are trying to backtrack. This means that all possible assignments under $x_{i_1 \ell(i_1)} = 1$ and $x_{i_1 \ell(i_1)} = 0$ have already been enumerated, i.e. all possible ways of assigning the m objects are enumerated, and we stop. The stored assignment and corresponding C* give the optimal solution.

Summary of the Algorithm

We have developed above all the details required to describe a specific solution procedure of the quadratic assignment problem.

Initialization Step

Let $P(i) = \emptyset$ for $i = 1,2,...,m$ (prohibited locations). Let $C^* = \infty$.

Choose an object i_1 and place it in location $\ell(i_1)$. i_1 can be determined by maximizing $\sum_{j=1}^{m} u_{ij}$ for $i = 1,2,...,m$, and $\ell(i_1)$ can be determined by minimizing $\sum_{i=1}^{n} d_{ij}$ for $j = 1,2,...,n$. Let $I = \{i_1\}$ and $J = \{j_1\}$. Let $k = 1$ and go to step 1.

Step 1 (Forward Move)

Calculate a lower bound B on all completions of current partial solution, $B = C_1 + C_2 + C_3$, where C_1, C_2, and C_3 are calculated as shown earlier, with the exception that $b_{ij} = \infty$ if $j \in P(i)$. If $B \geq C^*$ go to step 2. Otherwise pick $i_{k+1} \notin I$ such that $u_{i_{k+1} i_k} = \underset{i \notin I}{\text{Max}}\ u_{i i_k}$, and place i_{k+1} in $\ell(i_{k+1})$ where $\ell(i_{k+1})$ is determined by

$$\text{Minimizing}_{\substack{j \notin J \\ j \notin P(i_{k+1})}} \quad f_{i_{k+1} j} + \sum_{t=1}^{k} u_{i_{k+1} i_t} d_{j \ell(i_t)} + \sum_{t=1}^{k} u_{i_t i_{k+1}} d_{\ell(i_t) j}$$

Replace I by $I \cup \{i_{k+1}\}$ and J by $J \cup \{\ell(i_{k+1})\}$.

If $k = n-1$, calculate the new value of C_1 (cost of complete assignment). If $C_1 < C^*$, replace C^* by C_1, store $(i_t, \ell(i_t))$ for $t = 1,2,...,n$, replace k by n and go to step 2. If $C_1 \geq C^*$ then replace k by n and go to Step 2. If $k < n-1$, then replace k by $k+1$ and repeat Step 1.

Step 2 (Fathom)

i_k is removed from I and $\ell_{(i_k)}$ is removed from J. Replace $P(i_k)$ by $P(i_k) \cup \{\ell(i_k)\}$. Calculate a lower bound B on completions of the partial solution $x_{i_t \ell(i_t)} = 1$ for $t = 1,2,\ldots,k-1$ and $x_{i_k \ell(i_k)} = 0$, where $B = C_1 + C_2 + C_3$. C_1, C_2, and C_3 are calculated as mentioned above, with the exception that $b_{ij} = +\infty$ if $j \in P(i)$. If $B \geq C^*$ go to step 3. Otherwise assign i_k to a location $t \notin J \cup P(i_k)$ such that the cost $f_{i_k t} + \sum_{j=1}^{k-1} u_{i_k i_j} d_{t\ell(i_j)} + \sum_{j=1}^{k-1} u_{i_j i_k} d_{\ell(i_j)t}$ is minimized over $t \notin J \cup P(i_k)$. i_k is added to I, and t, the new $\ell_{(i_k)}$, is added to J. Go to step 1.

Step 3 (Strong Fathom)

i_{k-1} is deleted from I and $\ell_{(i_{k-1})}$ is deleted from J. $\ell(i_{k-1})$ is placed in $P(i_{k-1})$ and $P(i_k)$ is replaced by the empty set. k is replaced by k-1. If $k = 0$ go to step 4, otherwise go to step 2.

Step 4 (Termination)

Search has been completed. Optimal cost is C^* and its corresponding assignment $(i_k, \ell_{(i_k)})$, $k = 1,2,\ldots,n$ is the optimal assignment. Stop.

The following points are helpful to keep in mind.

1. During the initialization step an upper bound $C^* = \infty$ is used. As the search progresses, C^* denotes the objective of the best available complete assignment. Note that P(i) represents the locations that object i cannot be assigned to. These are initialized by the empty sets.
2. Step 1 represents a forward step, where the level of the tree increases by 1 unit. In this case the bound is less than C^*, and hence a complete solution with an objective better than C^* is possible.
3. Step 2 is a fathoming step, where $B \geq C^*$. In this case the last assignment is prohibited. Immediately a new bound is calculated. If the new bound is less than C^*, then a forward move is made. But if the bound is still greater than or equal to C^*, then a strong fathoming (Step 3) is made, and the level of the tree is reduced. Of course,

strong fathoming is most desirable, since it avoids the expensive task of trying to locate object i_k to a free location other than $\ell_{(i_k)}$.

Sub-Optimal Solutions

Due to its highly combinatorial nature, the task of finding and verifying the optimal solution within a reasonable computation time, for problems with $n > 15$, is almost impossible. In order to solve larger problems, we must resort to suboptimal solutions. Rather than using some heuristics, which may work very well for some problems, but which also may not be satisfactory for other problems, we will use the branch and bound procedure, itself, to obtain suboptimal solutions (or optimal) within a desired degree of accuracy from the optimal. We propose two methods for implementing this general strategy.

Method 1

Recall that a partial solution is fathomed, if the lower bound on all completions is at least as big as C*, the best known objective. Suppose that a partial solution is fathomed if $B \geq \alpha C^*$, where $\alpha \in (0,1]$. In this case, the partial solution is abandoned if there is no hope that it leads to an objective which is better than αC^*. To illustrate, suppose that C* is 1000, and we choose $\alpha = 0.95$. Then we fathom if $B \geq 950$. This means that if the bound on completions of the partial solution cannot lead to an improvement of 50 units over what we already have, then it does not pay off to pursue the partial solution. The objective of this simple strategy is clear. We want to fathom, i.e. abandon partial solutions, quickly, even if they lead to a slight improvement. As the computational experience shows, this simple strategy speeds the computational effort considerably. Of course, as a new C* is found, then we fathom whenever the bound is greater or equal to α times the new C*. The pro-

cedure continues until we cannot find a feasible solution with an objective less than αC^*. So we have a feasible assignment with objective C^*, coupled with the statement that the optimal objective is greater than or equal to αC^*.

Choice of α

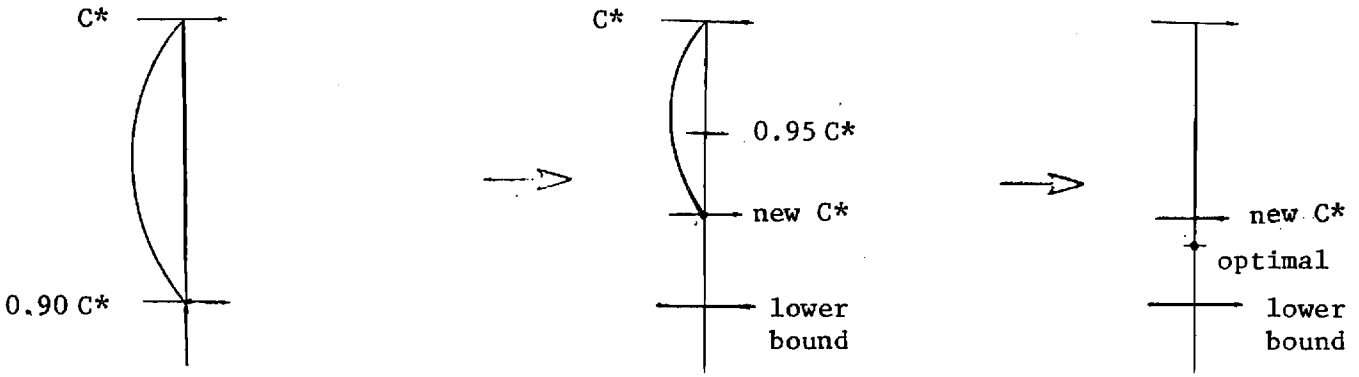
Of course, if α is small, then fathoming will speed up considerably, resulting in a small computational effort. But on the other hand, the quality of the best obtained feasible solution is not guaranteed. To illustrate, suppose that at termination C^* is 1000 and suppose that $\alpha = 0.60$. All we can say in this case is that we have a feasible solution with objective 1000, and the true optimal has an objective greater or equal to 600. This, of course, is not satisfactory. Suppose, however, that we used $\alpha = 0.95$. Then if at termination C^* is 1000, we can say that the optimal is greater than or equal to 950. This statement is indeed much stronger than the previous statement.

From this discussion, it is clear that there is a tradeoff between the computational effort and the value of α. We recommend values of $\alpha \geq 0.9$ depending on the accuracy required.

Exact Solution By Stepped Fathoming

A slight modification of the above scheme can be used to find the exact optimal solution. Suppose that an initial α_1 is used, say $\alpha_1 = 0.90$. Eventually we achieve a final C^* and cannot find a feasible solution with objective less than $0.90\,C^*$ i.e. $0.9\,C^*$ is a lower bound on all solutions. We either stop here, or else switch to $\alpha_2 \geq \alpha_1$, say $\alpha_2 = 0.95$. We start the search from the stored best solution corresponding to C^*, which also includes all the prohibited locations. We are either able to find a feasible solution with objective less than $0.95\,C^*$ (none less than $0.9\,C^*$ exist), and repeat the process, or else conclude that none exists. We may then

switch to 0.97 fathoming and eventually 100% fathoming. This of course, will lead to the optimal solution. Even though parts of the tree will be searched more than once, the quick fathoming outweighs some repeated efforts.



no solution with objective < 0.9 C* exists

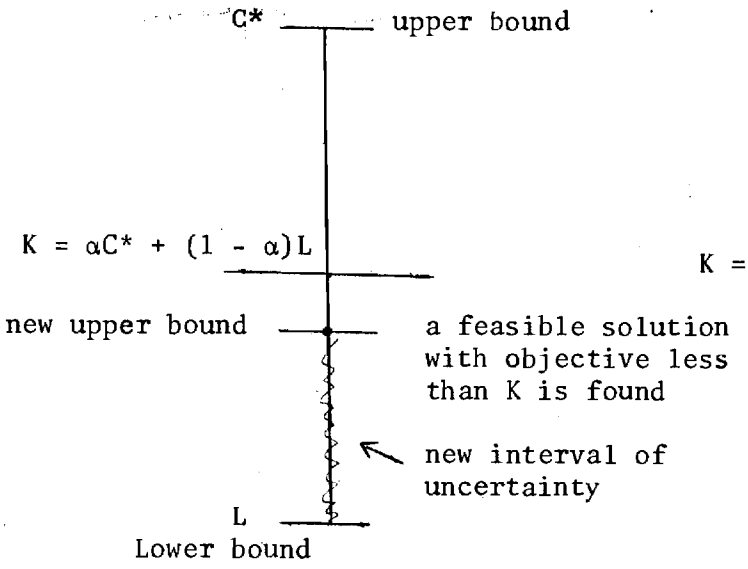
A solution with objective < 0.95 C* found

100% fathoming

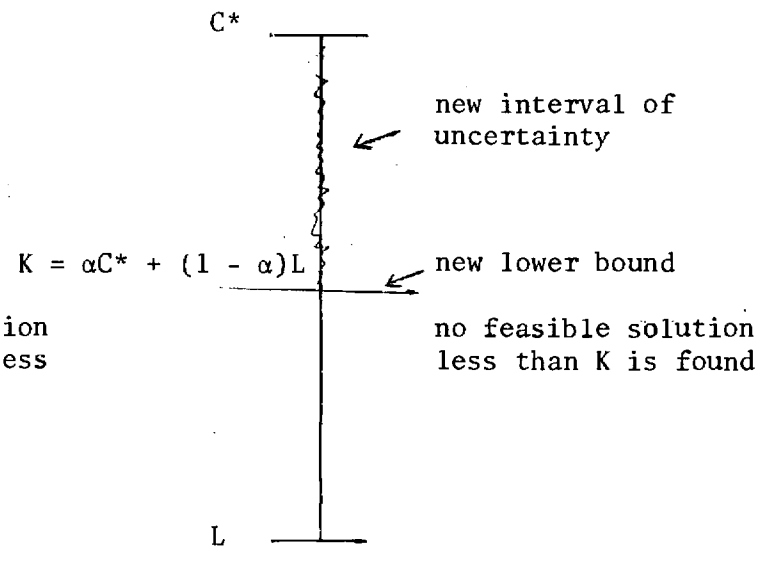
Method 2

This method again fathoms whenever the bound B is greater than or equal to K, where K is related to, but not necessarily equal to C* ($K \leq C^*$). Again, we have the same interpretation as before. A partial solution is pursued if it guarantees a nontrivial improvement over what we have got. We will describe a procedure to find a suitable K. At each stage of the algorithm, we have an upper bound (optimal $\leq$ upper bound), namely C*. A lower bound on the overall problem L, can be devised. At each stage of the algorithm, we have a feasible solution with objective C*, and we know that the optimal must be greater than or equal to C*. Rather than fathoming on C*, suppose we fathom on $K = \alpha C^* + (1 - \alpha)L$, where $\alpha \in [0,1]$. Since $L < C^*$, then $K = \alpha C^* + (1 - \alpha)L \leq C^*$. Two cases are possible. In the first case, we will be able to find a complete solution with objective less than K. The objective of this new solution becomes the new upper bound C* and the process is repeated. In the second case, we will not be able to find such a solution. This automatically implies

that there are no solutions with objective less than K, and hence K itself is the new lower bound. The process is repeated. From this we keep narrowing the gap between the lower and upper bounds, either by lowering the upper bound (finding an improved feasible assignment) or by raising the lower bound (by finding that there is no feasible solution with objective lower that the current lower bound). When the difference between the lower bound, and the current best objective value (upper bound) is smaller than a prescribed tolerance, we either stop, or switch the fathoming to $K = C^*$. Of course, if the tolerance is zero, or if we switch the fathoming to C^* (i.e., switch α to 1), then we will find the true optimal solution.



Upper bound reduced



Lower bound raised

Choice of α

α is any number in the interval [0,1]. Of course, if α is close to 1, then we are in effect fathoming on a number very close to C^*, and the search will not speed considerably. On the other hand, if α is close to zero, then improvement is achieved only if we obtain a feasible solution

very close to the overall lower bound. In this case, fathoming will be fast, but it is likely not to obtain feasible solutions less than K. So again a tradeoff is required here. We recommend values of α equal to 0.5 so that the interval of uncertainty is halfed at each time.

Calculation of an initial overall lower and upper bounds

An initial lower bound, and also a reasonable (not equal to ∞) initial upper bound are needed for the above procedure. To calculate the lower bound, first calculate a lower bound b_{ij} on the cost of locating object i to location j, for i = 1,2,...,m and j = 1,2,...,n. The interactions between object i and all other objects are put in a descending order, and similarly, the distances from location j and all other locations, are put in an ascending order. The inner product of the two vectors give b_{ij}. Then a linear assignment problem is solved to find L. More precisely, let L be the optimal objective of the following problem.

$$\text{Minimize} \quad \sum_{j=1}^{n} \sum_{i=1}^{m} b_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,2,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad j = 1,2,\ldots,n$$

$$x_{ij} \geq 0 \qquad \text{all i and j}$$

An upper bound C* is immediately available, by calculating the quadratic cost of the optimal assignment resulting from the above problem. Now Method 2 can be initiated.

Preassigned Objects

In many applications, a large number of objects is already preassigned, and only some new objects are to be placed, such that the overall cost

is minimized (e.g. some plants are already assigned, and it is desirable to assign some new plants). In this case, the algorithm can be applied with few obvious modifications in the calculations. Since the preassigned objects and their locations will always be assigned, then these objects will always be in the set I and their locations will always be in the set J. In the tree search, if γ objects are already assigned, we start the search by assigning more objects, i.e. the level of the tree starts at $\gamma+1$. If the level of the tree ever becomes γ, then we stop. If an overall lower bound on all feasible solutions to the problem is desired, we recommend calculating $C_1 + C_2 + C_3$ as before, where I and J are the indices of the preassigned objects and their locations.

A Heuristic For Fixing Locations Of Some Key Objects

For large problems, obtaining an optimal solution, or even a sub-optimal solution, with a good accuracy, may be prohibited due to the fact that the computational effort grows very fast as a function of the problem size. One of the difficulties of the quadratic assignment problem is that many solutions exist which are either optimal or very close to the optimal and hence, a large portion of the decision tree must be explicitly enumerated if the true optimal is to be found. This disadvantage can be significantly reduced by preassigning some objects in a seemingly good set of locations. These objects are then fixed once and for all, and the search continues on the remaining objects. If we choose the objects which are preassigned in a logical manner, and so their preassigned locations, even though we may miss the true optimal, it is likely that we will be very close to it, provided that the other objects are assigned optimally, once the preassigned objects are fixed. In a sense, the branch and bound procedure will tell us how to optimally assign the remaining objects, once we commit ourselves to preassigning some objects. If the

preassignment was done intelligently, we will likely be very close to the optimal solution, and meanwhile accomplish the following two tasks:

1. The tree level is reduced, thus drastically reducing the computational effort.
2. By fixing the location of some key objects, the lower bounds calculated will be tighter, which will allow faster fathoming, again resulting in reduced computational effort.

Basis for Choosing Preassigned Objects

Objects which interact with a large number of objects, and which have large total interactions, are desired to be assigned in "central" locations, so that they would be close to many possible objects. This remark can be used effectively to fix the locations of some objects. We can rank the objects according to a combination of:

1. The number of objects to which any given fixed object is interacting
2. The total number of interactions from any given object and all other objects

The locations are ranked according to their centralization, e.g. location i is ranked according to $\sum_{j=1}^{n} d_{ij}$. The best location i is the one with smallest $\sum_{j=1}^{n} d_{ij}$, and so forth. A possible way of fixing key objects is to assign the best ranking object to the best ranking location. Then the object which has the most number of interactions to the most recently assigned object is placed in a location which minimizes total cost. Then the next best ranking object can be assigned to a location which minimizes total cost, and so forth. This process is continued until a certain percentage of the objects are assigned, say 20%. We then seek improving the quality of the preassignment by rotating the positions of these objects. Then we go to the branch and bound scheme. This procedure has proved effective in obtaining good quality solutions.

Calculation Of Overall Lower Bounds In The Preassigned Case

We have discussed earlier the importance of fixing some objects to certain locations apriori. The branch and bound algorithm will then determine the optimal completion of the partial apriori fixed solution. We argued earlier, that if these fixed objects and their locations, are chosen in a logical manner, it is likely that the optimal not be missed by a large amount. It may be helpful, however, to calculate an overall lower bound, on the portion of the decision tree, which we did not enumerate, due to the initial partial solution.

To illustrate, suppose that object i is placed in location $\ell(i)$, once and for all during the computation, for i = 1,2,...,k. We would like to calculate a lower bound on the portion of the tree where not all the objects 1,2,...,k are assigned to $\ell(1)$, $\ell(2)$,..., $\ell(k)$. 2^k-1 cases must be investigated, since any given object i, can either be placed in $\ell(i)$ or at some other location. For each such case, a lower bound is calculated. An overall lower bound, on the unenumerated portion of the tree is then the minimal of these lower bounds. The lower bound, where object i is placed in $\ell(i)$ for $i \in I \subset \{1,2,...,k\}$, and object i is prohibited from location $\ell(i)$ for $i \in \{1,2,...,k\} - I$, can be calculated as follows.

$LB = C_1 + C_2 + C_3$

where

C_1 is the cost of interaction among objects in I.

$$C_1 = \sum_{i,j \in I} u_{ij} d_{\ell(i),\ell(j)} + \sum_{i \in I} f_{i\ell(i)}$$

C_2 is the cost of interaction from unassigned objects to objects in I

C_3 is the cost of interaction from unassigned objects to unassigned objects.

C_2 and C_3 can be calculated by solving the following linear assignment problem.

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j\notin J}}^{n} \sum_{\substack{i=1 \\ i\notin I}}^{m} b_{ij} x_{ij}$$

$$\text{Subject to:} \quad \sum_{\substack{j=1 \\ j\notin J}}^{n} x_{ij} = 1 \qquad \text{for } i \in \{1,2,\ldots,m\} - I$$

$$\sum_{\substack{i=1 \\ i\notin I}}^{m} x_{ij} \leq 1 \qquad \text{for } j \in \{1,2,\ldots,n\} - J$$

$$x_{ij} \geq 0 \qquad i \in \{1,2,\ldots,m\} - I, \quad j \in \{1,2,\ldots,n\} - J$$

where $J = \{ j : j = \ell(i) \text{ for some } i \in I \}$, and b_{ij} is a bound on the cost of assigning object i to location j. b_{ij} is calculated as follows.

$b_{i\ell(i)} = \infty$ if $i \in \{1,2,\ldots,k\} - I$

$b_{ij} = c_2 + c_3$ otherwise

where

$$c_2 = \sum_{t\in I} u_{it} d_{\ell(t)j} + \sum_{t\in I} u_{ti} d_{\ell(t)j} + f_{ij}$$

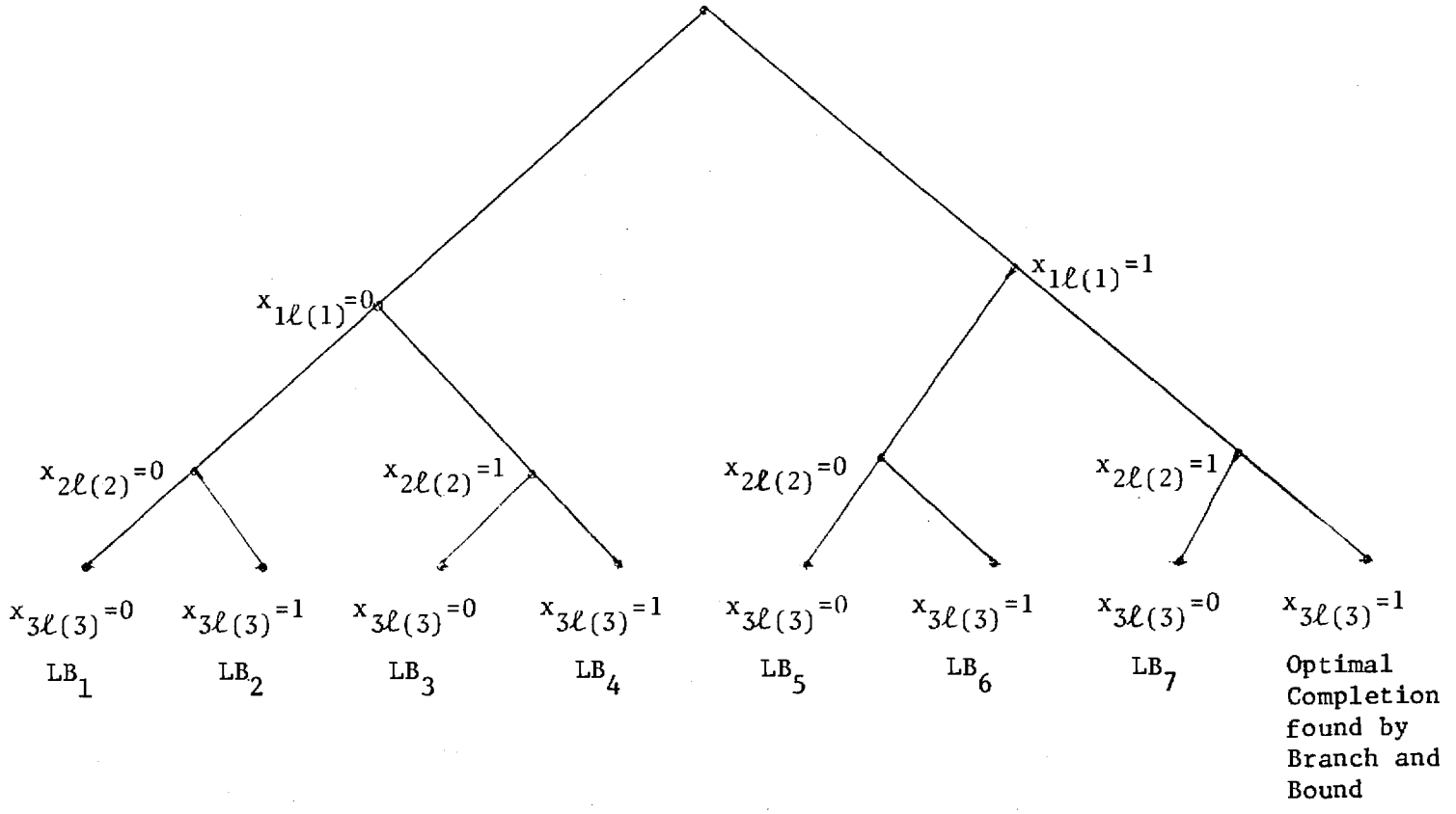
$$c_3 = v \cdot w$$

v is a vector of interactions from object i to objects $\{1,2,\ldots,m\} - I$, and from objects $\{1,2,\ldots,m\} - I$ to object i, arranged in descending order.

w is a vector of distances from location j to locations $\{1,2,\ldots,n\} - J$, and from locations $\{1,2,\ldots,n\} - J$ to location j, arranged in an ascending order. If $n \neq m$, complete the vector v by adding zeros from the bottom so that v and w have the same size.

After solving the above linear assignment problem, we get a bound on all solutions where the assignments i in $\ell(i)$ for $i \in I \subset \{1,2,\ldots,k\}$ is enforced, and the assignments i in $\ell(i)$ for $i \in \{1,2,\ldots,k\} - I$ are prohibited. Lower bounds on all possible ways of choosing $I \subset \{1,2,\ldots,k\}$ are calculated, and the minimal represents the absolute lower bound on the overall problem. The case of $k = 3$ is illustrated below. The optimal completion of $x_{i\ell(i)} = 1$ for $i = 1,2,3$ is

found by the algorithm. $LB_1, \ldots, LB_7$ are calculated as discussed above, and the overall lower bound on the unenumerated portion of the tree is then $\underset{1 \le j \le 7}{\text{minimum}} LB_j$.



Section III

THE TRAVELING SALESMAN PROBLEM

In this section we will present a branch and bound algorithm for solving nonsymmetric as well as symmetric traveling salesman problems. As a first step, we formulate and solve the dual of the traveling salesman problem. If the optimal dual solution results in an oriented tour (or just a tour in the symmetric case), then we also have an optimal solution of the traveling salesman problem. Otherwise, we at least obtain a right lower bound on the optimal solution of the traveling salesman problem, and a new distance matrix which is "well-conditioned". A branch and bound procedure, similar to that of the general quadratic assignment model, is devised to find the optimal tour, i.e., to close any gap between the primal and dual problems. The major simplification obtained for the special case of the traveling salesman problem is that we start the branch and bound search from an infeasible solution which is close to the optimal tour, thus considerably reducing the computational effort. Of course, this is made possible by the use of duality in obtaining a tight lower bound. The procedure, which is suitable for both symmetric and nonsymmetric traveling salesman problems, is an extension of the method of Held and Karp for solving the symmetric problem.

Formulation of the Problem and its Dual

Suppose that a salesman is given a list of n cities, where each pair of cities, i and j, is connected by a link of length c_{ij} (if cities i and j are not directly linked, let $c_{ij} = +\infty$). The problem is then to find

a tour of minimal length which the salesman can follow so as to visit each city exactly once, and then return to where he started.

Let x_{ij} be a variable relating to link ij. x_{ij} assumes a value of 1 or a value of 0. If the salesman travels from city i to city j, then $x_{ij} = 1$, otherwise $x_{ij} = 0$. The problem can then be stated as follows:

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} c_{ij} x_{ij}$$

$$\text{Subject to:} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} = 1 \qquad i = 1, 2, 3, \ldots, n \qquad (1)$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1 \qquad j = 1, 2, 3, \ldots, n \qquad (2)$$

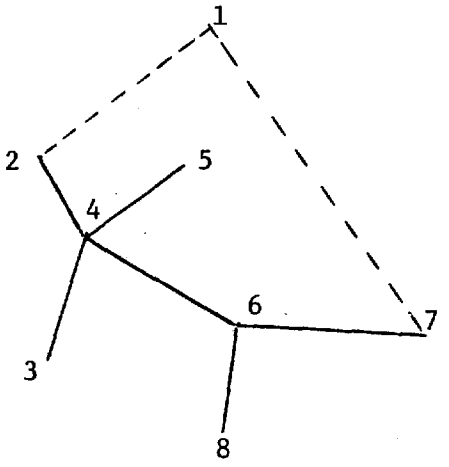
$$x_{ij} \text{ is 0 or 1} \qquad i = 1, \ldots, n;\ j = 1, \ldots, n \qquad (3)$$

No subtours

Since the restrictions (1), (2), and (3), by themselves, may allow the meaningless answer of having two or more disjoint subtours, a restriction eliminating subtours must be explicitly introduced. There are various methods in the literature for enforcing this restriction. We will adopt the one-tree method of Held and Karp, which if coupled with (1), (2), and (3), will ensure the elimination of subtours.

Let us begin by reviewing the definition of a 1-tree [10]. A 1-tree of the graph composed of the n cities and their interconnecting links is a tree which spans $n-1$ cities, coupled with two distinct links from the remaining city, to any two of the $n-1$ cities spanned by the tree. Therefore, by definition, a 1-tree contains exactly one subtour, or one

cycle. Of course, the number of links in the cycle is less than or equal to n. An example of a 1-tree is illustrated below. Notice that the 1-tree is composed of a tree spanning cities (nodes) 2 through 8, plus two links from node 1 to nodes 2 and 7.



An Example of a 1-Tree

Now recall that constraints (1), (2), and (3) either admits no subtours, or else at least two subtours. By definitions of the one-tree, it admits exactly one subtour (which may be a complete tour if it has exactly n links). Therefore constraints (1), (2), and (3), coupled with the constraint that the vector x forms a 1-tree, admit no subtours. Therefore, the traveling salesman problem can be formulated as follows:

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} c_{ij} x_{ij}$$

$$\text{Subject to:} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} = 1 \qquad i = 1, 2, \ldots, n$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

$$x \in X$$

where X is the set of 1-trees spanning the graph, i.e.,

$$X = \{(x_{11}, \ldots, x_{1n}, \ldots\ x_{n1}, \ldots, x_{nn}) : x_{ij} = 0 \text{ or } x_{ij} = 1\ (i \neq j),$$
$$\text{and the links ij with } x_{ij} = 1 \text{ form a 1-tree}\}.$$

Note that there are very efficient procedures for finding a 1-tree with minimal cost [5, 13]. This will involve pulling one node out of the n nodes, which we will refer to as the initial node, and then:

1. finding a tree of minimal length which spans the remaining $n-1$ nodes
2. finding two links of minimal length which connect the initial node to the spanning tree.

Any node can be used as the inital node, and we will discuss later a procedure for making a "good" choice of the initial node. The calculation of the 1-tree with minimal cost, in the presence of restrictions (1), (2), and (3), namely the traveling salesman problem, is indeed a difficult problem. If we handle the restrictions (1) and (2) via Lagrangian multipliers (dual variables) and put them into the objective function, we are left with the very simple problem of finding a minimal 1-tree. The results of the minimal 1-tree can be used to update the Lagrangian multipliers. This leads to the following dual formulation.

Dual Problem

Maximize $\theta(u,v)$

u and v are unrestricted

where,

$$\theta(u,v) = \underset{x \in X}{\text{Minimum}} \left\{ \sum_j \sum_i c_{ij} x_{ij} + \sum_i u_i \left(\sum_j x_{ij} - 1 \right) + \sum_j v_j \left(\sum_i x_{ij} - 1 \right) \right\}$$

$$= \underset{x \in X}{\text{Minimum}} \left\{ \sum_j \sum_i (c_{ij} + u_i + v_j) x_{ij} - \sum_i u_i - \sum_j v_j \right\}$$

$$= -\sum_i u_i - \sum_j v_j + \underset{x \in X}{\text{Minimum}} \left\{ \sum_j \sum_i (c_{ij} + u_i + v_j) x_{ij} \right\}$$

Note that evaluating $\theta(u,v)$, for fixed values of u and v is an easy task since it involves finding a minimal 1-tree, where the cost c_{ij} of the link ij is replaced by $c_{ij} + u_i + v_j$. Also note that in finding the minimal 1-tree, we are not concerned about the orientation of the links. In other words we are actually searching for a minimal 1-tree, rather than an aborescence. This simplifies the problem of evaluating $\theta(u,v)$ considerably. Of course, we may think of the vectors u and v as penalties for violating the proper orientation of the links. So if when calculating the minimal 1-tree, too many links "left" node i, we will attempt to "penalize" this deficiency in the next iteration by increasing u_i. Similarly, if too many links "entered" node j, then we will attempt to "penalize" this deficiency by increasing v_j.

Simplification of the Symmetric Case

In the symmetric case, as long as the vector x forms a 1-tree, and since the orientation of links is not important, we no longer require that the number of links leaving node i is 1, and the number of links entering node i is 1, so long as the number of links incident to node i is equal to 2. The primal problem can thus be reformulated as follows:

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{i=1}^{n} c_{ij} x_{ij}$$

$$\text{Subject to:} \quad \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} + \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ji} = z \qquad \text{for } i = 1, 2, \ldots, n$$

$$x \in X$$

where X is the collection of 1-trees. The above restrictions ensure

forming a loop. If the loop is not "oriented", it can be made so, by simply reorienting some of the links, without affecting the cost. The dual of the above problem is given below, with only one set of dual variables given by the vector u.

Maximize $\quad \theta(u)$

u is unrestricted

where,

$$\theta(u) = \underset{x \in X}{\text{Minimum}} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{n} u_i \left(\sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} + \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ji} - 2 \right)$$

$$= -2 \sum_{i=1}^{n} u_i + \underset{x \in X}{\text{Minimum}} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} (c_{ij} + u_i + u_j) x_{ij}$$

From now on we will concentrate on the general nonsymmetric case with dual variables u and v. For the symmetric case, the dual problem can be simplified as illustrated above, with no need for the dual variable v.

Interpretation of the Dual Problem

Let $f(x) = cx$, $g(x)$ and $h(x)$ be n-vector functions defined as follows:

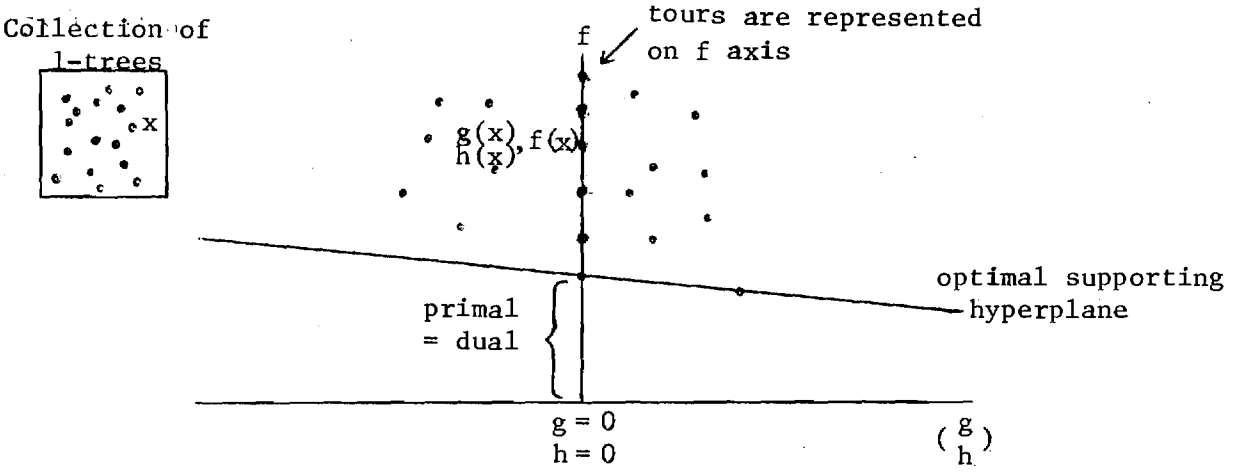
$$g_i(x) = \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} - 1 \quad \text{and} \quad h_i(x) = \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ji} - 1 \quad \text{for } i = 1, 2, \ldots, n.$$

The dual problem can be interpreted on the $\left(\binom{g}{h}, f\right)$ plane. Each 1-tree in X can be mapped into the $\left(\binom{g}{h}, f\right)$ plane as $\left(\binom{g(x)}{h(x)}, cx\right)$. Doing this for all 1-trees, we can construct the image of X in the $\left(\binom{g}{h}, f\right)$ plane, which will be composed of discrete points. Since a feasible tour is a 1-tree with $g(x) = h(x) = 0$, then all feasible tours lie on the f axis, and the minimal feasible tour is the one which is "lowest" on the f axis. Now given (u,v), $\theta(u,v) = \underset{x \in X}{\text{Minimum}} \; cx + ug(x) + vh(x)$. Therefore, evaluating θ corresponds to finding a hyperplane with slope $\binom{-u}{-v}$

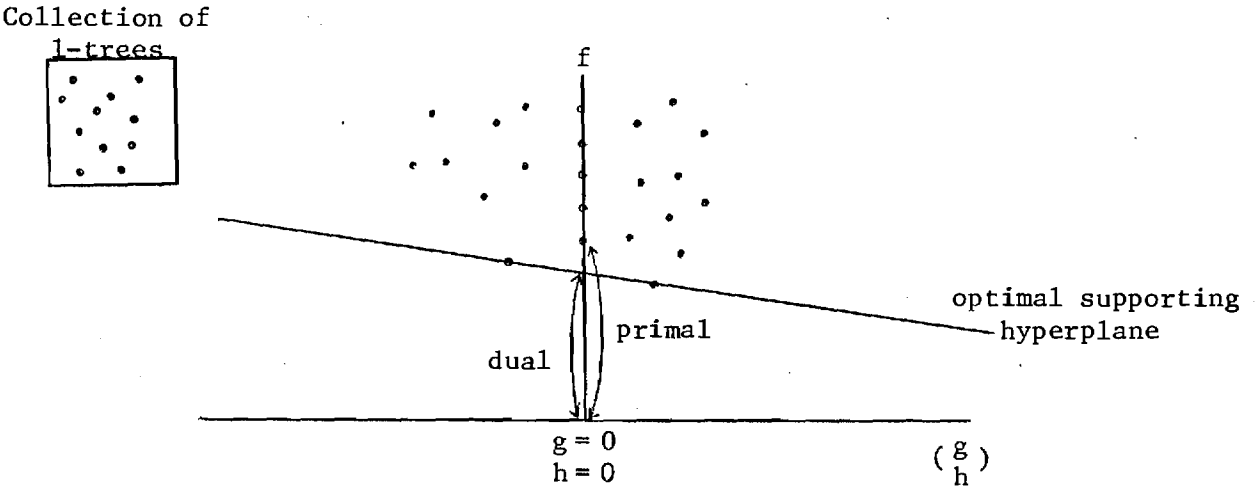
which supports the set of discrete points in the $\left(\binom{g}{h}, f\right)$ plane, and which has minimal intercept on the f axis. Therefore, the dual problem can be stated as follows:

Find the hyperplane which supports the image of X in the $\left(\binom{g}{h}, f\right)$ plane whose minimal intercept on the f axis is maximal.

Two cases are possible. In the first case, the optimal hyperplane supports the set at a point on the f axis, i.e., at the optimal primal tour, in which case there is no duality gaps. In the second case, the optimal supporting hyperplane does not pass through the optimal tour, in which case a gap exists.



Case of No Duality Gap



Case of A Duality Gap

Some Properties of the Dual Function θ

We will now study the dual function θ. It turns out that θ is concave and piecewise linear in u and v. This will help us determine how to modify the dual variables in such a way to maximize θ.

Let the collection of 1-trees, with a fixed initial node, be $x_1, x_2, \ldots, x_t$. Then $\theta(u,v)$ can be written as follows:

$$\theta(u,v) = \underset{x \in X}{\text{Minimum}} \quad \sum_i \sum_j c_{ij}x_{ij} + \sum_i u_i \left(\sum_j x_{ij} - 1 \right) + \sum_j v_j \left(\sum_i x_{ij} - 1 \right)$$

$$= \underset{x \in X}{\text{Minimum}} \quad cx + u\,g(x) + v\,h(x)$$

$$= \underset{1 \le j \le t}{\text{Minimum}} \quad cx_j + u\,g(x_j) + v\,h(x_j)$$

It is clear that θ is the minimum of a finite number of affine functions in u and v, namely $cx_j + u\,g(x_j) + v\,h(x_j)$ for $j = 1, 2, \ldots, t$. The dual function is illustrated below, and its concavity and piecewise linearity is clear from its construction. Note that each of the hyperplanes defining θ corresponds to a 1-tree.

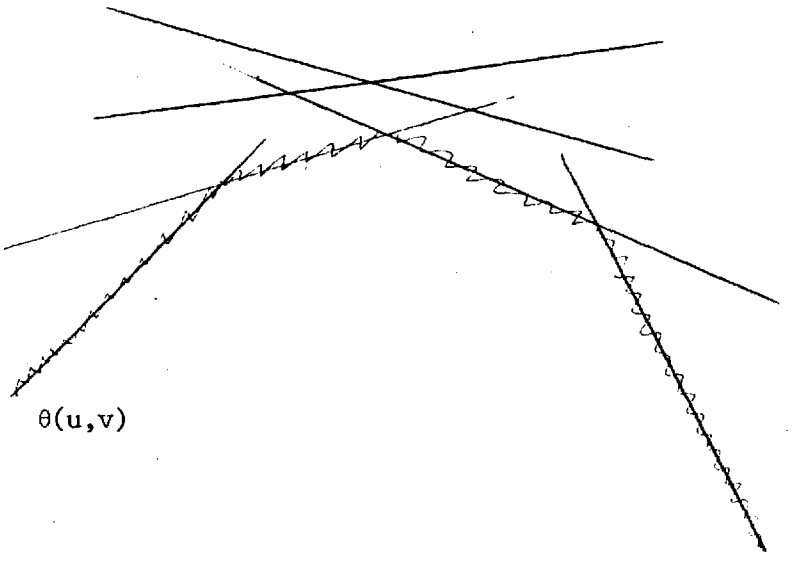
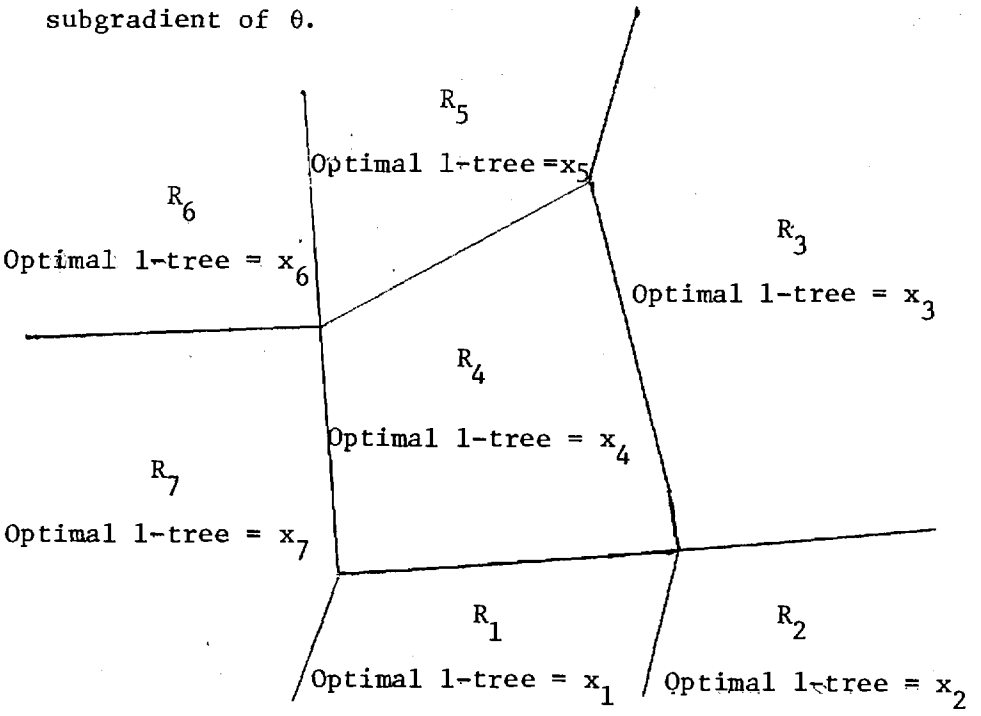


Illustration of the Dual Function

If we examine the (u,v) space, then it will be divided in regions $R_1, R_2, \ldots, R_t$, where in region R_j, $\theta(u,v) = cx_j + u\,g(x_j) + v\,h(x_j)$. Note that some regions may be empty, which means that the corresponding 1-tree is never minimal regardless of u and v. At any rate, each R_j in a polyhedral set, and for any (u,v) in the intersection of R_i and R_j, we have,

$$\theta(u,v) = cx_i + u\,g(x_i) + v\,h(x_i) = cx_j + u\,g(x_j) + v\,h(x_j)$$

Also, note that in the interior of region R_j, θ is differentiable (actually linear) and has gradient $\begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$. θ is not differentiable at the intersection of more than one region, but is subdifferentiable there, and the subgradients there can be characterized as convex combinations of the gradients in these regions. In particular, if $(u,v) \in \bigcap_{j \in I} R_j$, then θ is subdifferentiable at (u,v) and any subgradient is characterized by $\sum_{j \in I} \lambda_j \begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$ where $\sum_{j \in I} \lambda_j = 1$ and $\lambda_j \geq 0$ for each $j \in I$. In particular, each $\begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$ for $j \in I$ is a subgradient of θ.



Regions in the (u,v) Space

Some Strategies for Maximizing θ

There are various strategies for maximizing the dual function θ, which makes good use of its special structure. We will discuss some alternative possibilities. Of course, the following procedures optimize the dual, but not necessarily the primal. In the absence of a duality gap, the optimal tour will be obtained in the process. Otherwise, as we will discuss later, a branch and bound scheme will be used. We summarize below three strategies for maximizing the dual function θ. This is then followed by a more detailed discussion of each procedure.

Strategy 1 Tangential Approximation (or Decomposition)

The dual function θ is approximated by only a portion of the affine functions defining it, and at each iteration the optimal (u,v) of the "tangential approximation" is found. Finding the shortest 1-tree either ends in verifying optimality or in adding another hyperplane, resulting in a better approximation of θ.

Strategy 2 Steepest Ascent Optimization

At each point (u,v), the steepest ascent direction of θ is first found, then we optimize along this direction.

Strategy 3 Subgradient Optimization

The dual function is maximized by moving along subgradient vectors. In most cases this results in improving θ, and/or getting closer to the optimal point. At each iteration a new subgradient is found by finding the shortest 1-tree.

Strategy 1 Tangential Approximation (or Decomposition)

The decomposition principle can be used for maximizing the dual function. It may be helpful the rewrite the dual problem in a linear programming format, where z replaces $\theta(u,v)$.

Dual Problem

Maximize z

Subject to: $z \leq cx_j + u\,g(x_j) + v\,g(x_j)$ for $j = 1, 2, \ldots, t$

z, u, v unrestricted in sign

The above dual problem is equivalent to its own dual. Denoting the dual variables by λ_j, the above problem can be alternatively represented as follows:

Minimize $\sum_{j=1}^{t} \lambda_j (cx_j)$

Subject to: $\sum_{j=1}^{t} \lambda_j g(x_j) = 0$

$\sum_{j=1}^{t} \lambda_j h(x_j) = 0$

$\sum_{j=1}^{t} \lambda_j = 1$

$\lambda_j \geq 0 \qquad j = 1, 2, \ldots, t$

Since the number of 1-trees is very large, the dual problem has too many rows, and, needless to say, its own dual, has too many columns. Hence, the decomposition principle can itself be used as a row generation, or column generation technique for solving the dual problem. In particular, suppose that only k 1-trees, say $x_1, \ldots, x_k$ are available. Then we can solve a master problem, which generates new dual variables (u_k, v_k).

These in turn are passed to a subproblem which finds a new 1-tree. An optimality test can be performed to test whether the optimal solution to the dual problem has been obtained. The following steps are typical of such an algorithm.

Initialization Step

Find an initial feasible tour; a suitable heuristic can be used to find such a solution. This tour is x_1. Let $k = 1$ and go to Step 1.

Step 1 (Master Problem)

Solve the following problem.

$$\text{Minimize} \quad \sum_{j=1}^{k} \lambda_j (cx_j)$$

$$\text{Subject to:} \quad \sum_{j=1}^{k} \lambda_j g(x_j) = 0$$

$$\sum_{j=1}^{k} \lambda_j h(x_j) = 0$$

$$\sum_{j=1}^{k} \lambda_j = 1$$

$$\lambda_j \geq 0 \quad \text{for } j = 1, 2, \ldots, k$$

Let $z_k = \sum_{j=1}^{k} \lambda_j (cx_j)$, and let u_k and v_k be the dual variables of the first two sets of constraints, respectively (the dual of the above problem can be solved, but the above problem is used for computational convenience).

Step 2 (Subproblem)

Find the 1-tree of minimal length by solving the following subproblem.

$$\underset{x \in X}{\text{Minimize}} \quad (cx + u_k g(x) + v_k h(x))$$

Let the optimal be x_{k+1}. Go to Step 3.

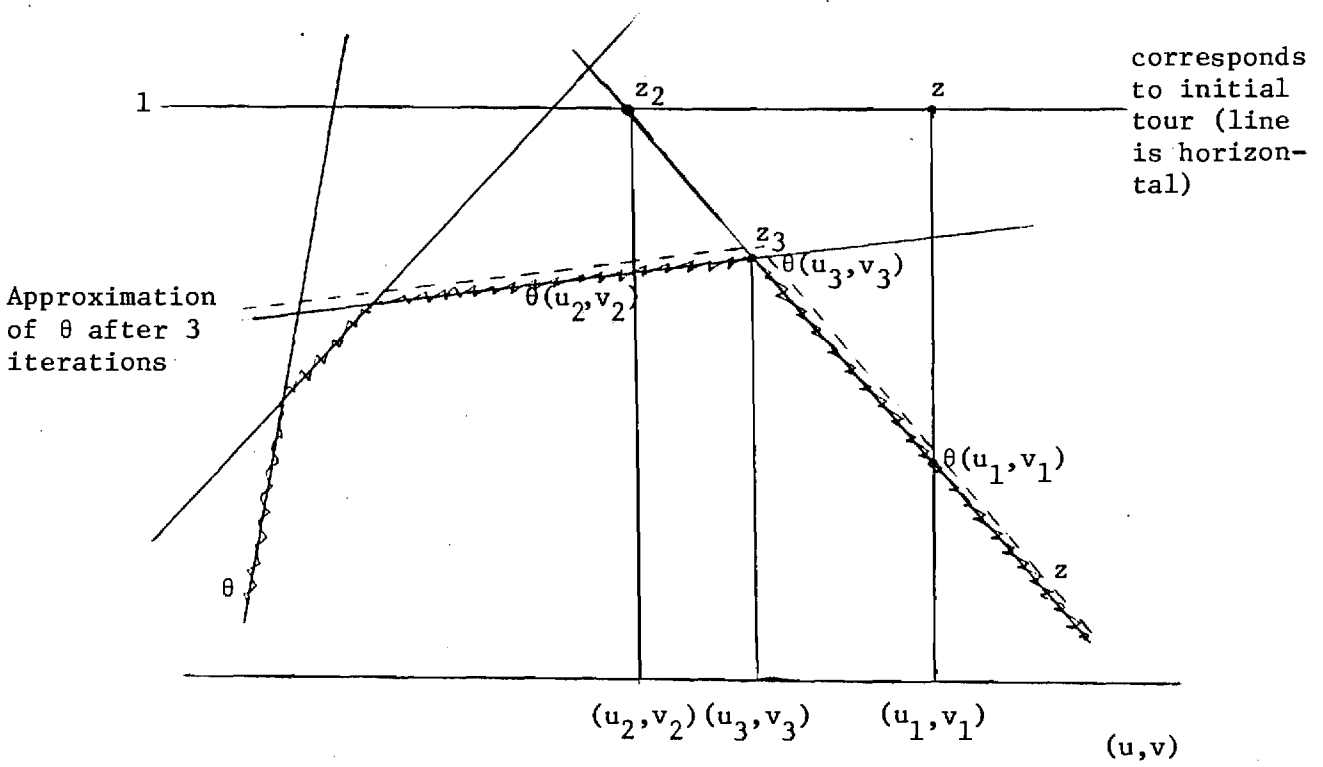
Step 3 (Optimality Test)

If $z_k = cx_{k+1} + u_k g(x_{k+1}) + v_k h(x_{u+1})$ then the optimal of the dual problem is z_k and we stop. Otherwise replace k by k+1 and go to Step 1. (Update the column $\begin{pmatrix} g(x_{k+1}) \\ h(x_{k+1}) \end{pmatrix}$ by multiplying it by the inverse basis of the optimal tableau of Step 1, and put the updated column in the optimal tableau of Step 1.)

Of course, if the optimal 1-tree at any stage is a tour, then we have already obtained the optimal solutions of the primal and dual problems, and we stop. If the "ideal" case is not realized, rather than solving the dual problem exactly, we may terminate whenever the current z_k is close enough to the best known lower bound up to iteration k, namely, $\underset{1 \le j \le k}{\text{maximum}} \theta(u_j, z_j)$.

Finiteness of the above procedure is obvious, since it is precisely the Dantzig-Wolfe decomposition (see, for example, [3,4,14]) applied to the above problem. The above procedure can also be interpreted in many other ways, e.g., as a cutting plane procedure [17], grid optimization [14], and tangential approximation [8,14]. We will interpret the above procedure as a tangential approximation method. This will give a distinct flavor to this strategy, which is significantly different from strategies 2 and 3, which are indeed very similar to each other. Recall that the master problem optimizes an "approximation" of θ, namely, the minimum of $cx_j + u\,g(x_j) + v\,g(x_j)$ for $j = 1, 2, \ldots, k$. So the master problem indeed finds an optimal, not to the overall prob-

lem, but to an approximation of it. Of course, if k = t, then the approximation is perfect (not necessarily conversely). After that, a decision as to whether the approximation is good enough, that is, whether the optimal to the approximation of θ is also the optimal of θ, must be made. This is done via the subproblem. The value of θ at the optimal dual variables of the approximation problem is evaluated by a simple minimization of a 1-tree problem. If $z_k \leq cx_{k+1} + u_k g(x_{k+1}) + v_k h(x_{k+1})$ where (z_k, u_k, v_k) solve the master problem, x_{k+1} is the optimal of the subproblem; then we must stop. Otherwise, we conclude that the current approximation is not satisfactory, and the new cut, $z \leq cx_{k+1} + u\, g(x_{k+1}) + v\, h(x_{k+1})$ is added to the master problem. This procedure is illustrated below. Note that the search is preferred to start with a complete tour, i.e., $g(x_1) = 0$ and $h(x_1) = 0$, otherwise the optimal of the first master problem is $+\infty$. At any rate, at each iteration, a better approximation of θ is realized. At iteration 1, z_1 is the cost of the initial feasible tour. Note that $z_1 > \theta(u_1,v_1)$, so the cut $z \leq cx_1 + u\, g(x_1) + v\, h(x_1)$ is added. In the second iteration, $z_2 > \theta(u_2,v_2)$, and hence, another cut is added. At iteration 3, $z_3 = \theta(u_3,v_3)$, and we stop. Of course, the z_j's are monotone decreasing. $\theta(u_j,v_j)$'s form lower bounds but are not necessarily monotone. Hence, as the procedure continues, we are not necessarily improving each (u_j,v_j), but rather improving the approximation of θ, and hence, z.



Tangential Approximation of θ

Strategy 2 (Steepest Ascent Optimization)

Strategies 2 and 3 improve the value of θ until optimality is reached. So, given (u,v), we attempt to find a direction such that while moving along it we will increase the value of θ. This distinguishes strategies 2 and 3 from strategy 1, which completely optimizes an "approximation" of θ at each stage, and hence does not necessarily improve θ itself at every iteration. Since θ is concave piecewise linear, finding the direction of steepest ascent "looks" promising. Before we discuss in more detail how to find the direction of steepest ascent, we will present precise definitions of ascent and steepest ascent directions.

Definitions

A vector (u,v) with norm 1 is called a direction of ascent of θ at (u_k,v_k) if $\theta'\left[(u_k,v_k);(u,v)\right] = \lim_{\lambda\to 0^+} \dfrac{\theta(u_k + \lambda u,\ v_k + \lambda v) - \theta(u_k,v_k)}{\lambda} > 0$.

A direction of ascent (u^*,v^*) is called a direction of steepest ascent if $\theta'\left[(u_k,v_k);(u^*,v^*)\right] = \underset{\|(u,v)\|=1}{\text{Maximum}}\ \theta'\left[(u_k,v_k);(u,v)\right]$.

If θ has no directions of ascent at (u_k,v_k), then the steepest ascent at (u_k,v_k) is called zero. By concavity of θ, it is clear that θ attains its maximum at (u_k,v_k) if the steepest ascent there is zero.

Characterization of Ascent and Steepest Ascent Directions

Suppose that (u_k,v_k) is given, with $\theta(u_k,v_k) = cx_j + u_k g(x_j) + v_k g(x_j)$ for $j \in I$. Note that $\theta'\left[(u_k,v_k);(u,v)\right] = \lim_{\lambda\to 0^+} \dfrac{\theta(u_k + \lambda u,\ v_k + \lambda v) - \theta(u_k,v_k)}{\lambda}$

$= \underset{\zeta\ \in\ E}{\text{Minimum}} \left\langle \zeta, (u,v) \right\rangle$, where E is the collection of subgradients of θ at (u_k,v_k). (See, for example, Rockafellar [15] and Grinold [9]).

But $\zeta \in E$ can be characterized as:

$$\sum_{j\in I} \lambda_j \begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$$

$$\sum_{j\in I} \lambda_j = 1$$

$$\lambda_j \geq 0 \quad \text{for } j \in I .$$

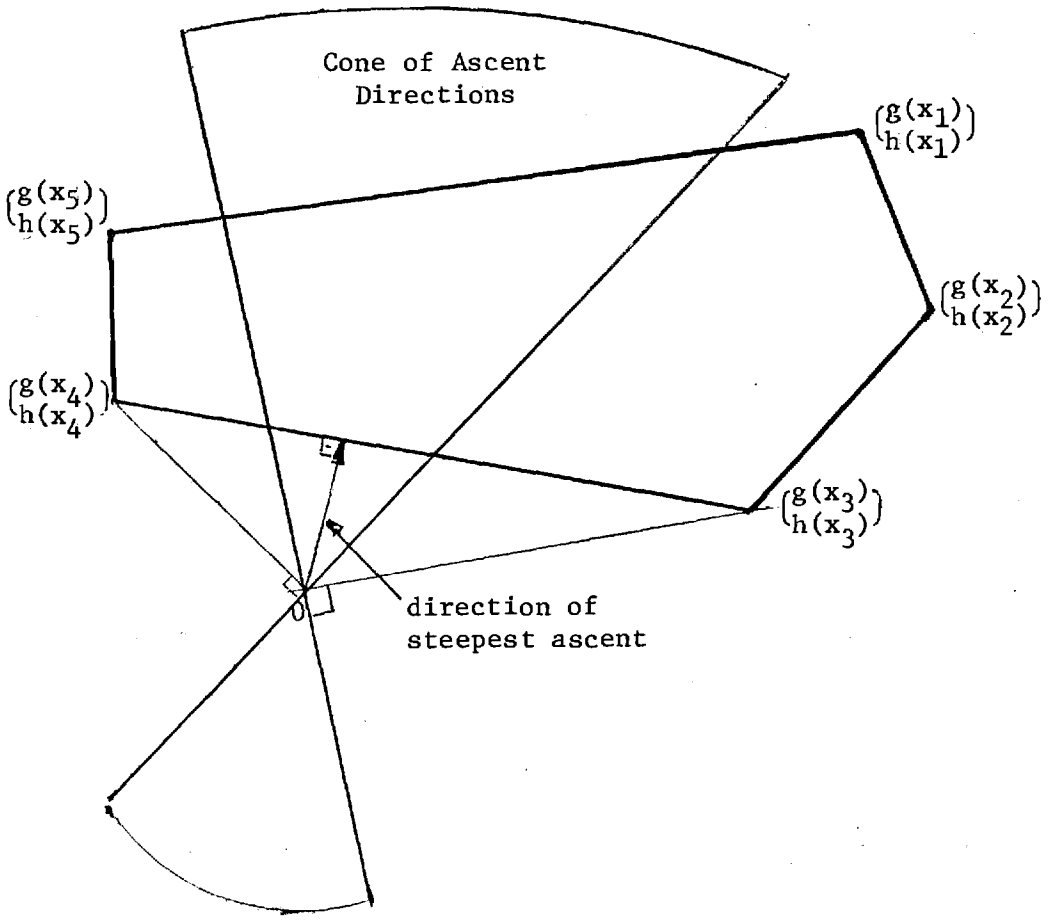
Therefore, (u,v) is an ascent direction if and only if $u\,g(x_j) + v\,h(x_j) > 0$ for each $j \in I$. A steepest ascent direction can further be characterized as follows:

$$\begin{aligned} \theta'\left[(u_k,v_k);(u^*,v^*)\right] &= \underset{\|(u,v)\|=1}{\text{Maximum}}\ \theta'\left[(u_k,v_k);(u,v)\right] \\ &= \underset{\|(u,v)\|=1}{\text{Maximum}}\ \underset{\zeta\ \in\ E}{\text{Minimum}} \left\langle \zeta, (u,v) \right\rangle \\ &= \underset{\zeta\ \in\ E}{\text{Minimum}}\ \underset{\|(u,v)\|=1}{\text{Maximum}} \left\langle \zeta, (u,v) \right\rangle \end{aligned}$$

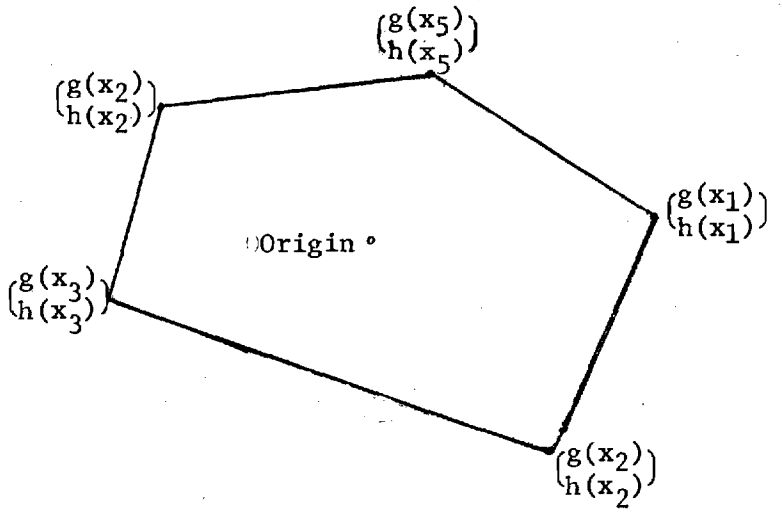
where the Max Min is replaced by Min Max due to linearity in ζ and (u,v) and compactness of E and the set $\{(u,v) : \|(u,v)\| = 1\}$. (See, for example, [ ].) If $\zeta \neq 0$ then the Maximum $\langle \zeta, (u,v) \rangle$ is given by letting $(u,v) = \frac{\zeta}{\|\zeta\|}$. Hence, $\theta'\left((u_k,v_k);(u^*,v^*)\right) = \underset{\zeta \in E}{\text{Minimum}} \|\zeta\|$. In particular, the direction of steepest ascent is then the subgradient with minimum length, normalized by dividing by its norm.

Geometric Interpretation

Given a point (u_k,v_k) where $\theta(u_k,v_k) = cx_j + u_k g(x_j) + v_k h(x_j)$ for $j \in I$, there is an interesting geometric interpretation of ascent and steepest ascent directions. Consider the lattic points $\begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$ for $j \in I$ in the (u,v) space. The convex combinations of these points is a polyhedron and forms the set of subgradients of θ at (u_k,v_k). The lattice points $\begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$ for $j \in I$ are the extreme points of this polyhedron. Not all points in the polyhedron are ascent directions. (u,v) is an ascent direction if and only if $u\,g(x_j) + v\,h(x_j) > 0$ for each $j \in I$, that is, if it belongs to the cone shown below ($I = \{1,2,3,4,5\}$). In particular, the steepest ascent direction is the subgradient (normalized so that it has norm 1) closest to the origin. In particular, if the polyhedron contains the origin, then the steepest ascent is zero, and (u_k,v_k) is the maximum of θ. The process of finding the steepest ascent has actually been used in many contexts. The above development gives a geometric characterization of these directions. In fact, the infeasiblity decomposition procedure of Balas [2], the procedure of Grinold [9], and more recently the procedure of Fisher, Northup, and Shapiro [6] can be viewed of methods for finding the steepest ascent



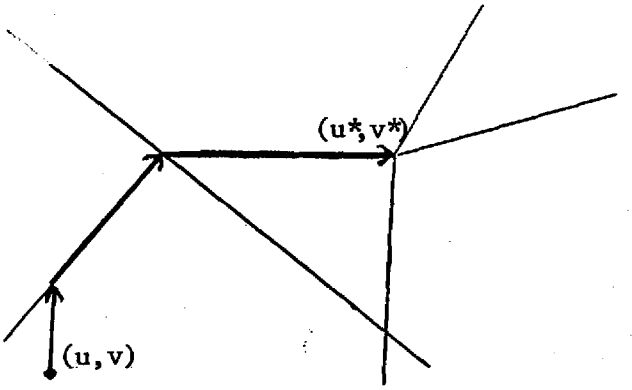
Directions of Ascent and Steepest Ascent



Steepest Ascent is Zero

direction with the ℓ_1-norm so that linear programming can be used. In Bazaraa and Goode [1], a procedure for finding the steepest ascent with Euclidean norm is presented.

We will describe some of the advantages and disadvantages of moving along the steepest ascent direction. The advantage of the steepest ascent optimization is obvious. We move along the best local optimizing direction, either across regions, or along ridges, until the optimal is reached in a finite number of steps, as illustrated below.



Moving Along Steepest Ascent Directions

The disadvantage of a steepest ascent procedure is that finding the steepest ascent direction, if we are in the intersection of many regions, is not an easy task. Before the shortest convex combination of $\begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$ for $j \in I$ is found, we must find $\begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$ for $j \in I$, either at once, or through a generic step, which may not be a trivial matter in this case.

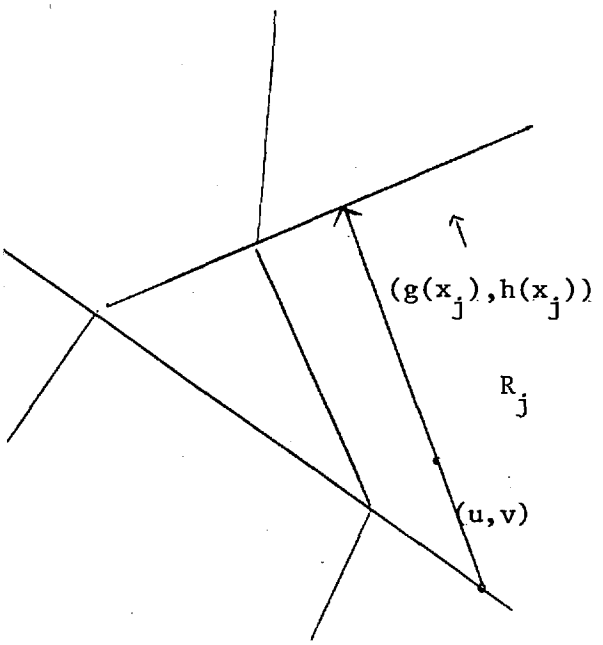
Finally, even if we find the direction of steepest ascent, finding how far to move along it, is not trivial, since we have to find the point

at which a new 1-tree becomes an alternative optimal (an LP based calculation may prove useful here. Of course, an approximation through utilization of discrete steps may be used, as in Held and Karp [11] (also see [12]) in their subgradient optimization method of the symmetric traveling salesman problem.

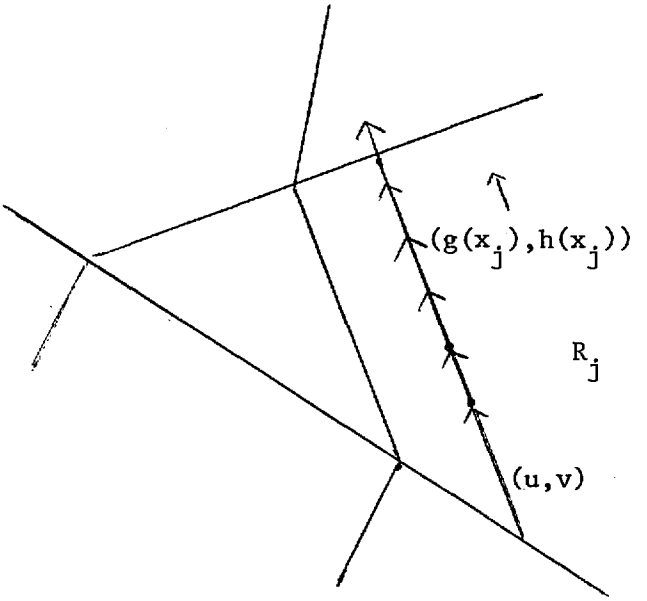
Strategy 3 (Subgradient Optimization)

Rather than moving along steepest ascent directions, one can move along subgradients, and in particular, along extreme points of the polyhedron formed by the subgradients, namely, $\begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$ for $j \in I$. This procedure has been successfully used by Held and Karp [11].

Of course, if we are at a point (u,v) which belongs to the interior of a region R_j, i.e., if there exists a unique minimal 1-tree where the cost of link ij is $c_{ij} + u_i + v_j$, then θ is differentiable at (u,v) and the unique subgradient is $\begin{pmatrix} g(x_j) \\ h(x_j) \end{pmatrix}$, and moving along it a positive distance will increase θ. Of course, the question is how far to move along this direction. Needless to say, $\theta\big((u,v) + \lambda(g(x_j),h(x_j))\big)$ is an increasing function of λ as long as $(u,v) + \lambda(g(x_j),h(x_j)) \in R_j$. Rather than finding the value of λ at which $(u,v) + \lambda(g(x_j),h(x_j))$ hits the boundary of R_j, i.e., when an alternative minimal 1-tree appears, one may move a small discrete step along $(g(x_j),h(x_j))$, and then calculate θ at the new point. If the minimal 1-tree is still x_j, then we still move a discrete step in the direction $(g(x_j),h(x_j))$ and repeat the process. Here there is a tradeoff between the computational effort spent in finding how far to move before leaving R_j, and the computational effort in repeating the minimal 1-tree calculation several times, one every time a discrete step is taken.



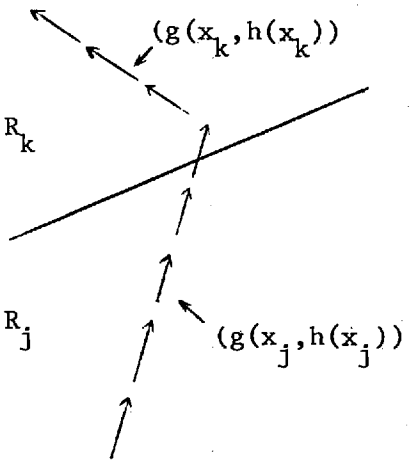
Moving until the Boundary of R_j is reached



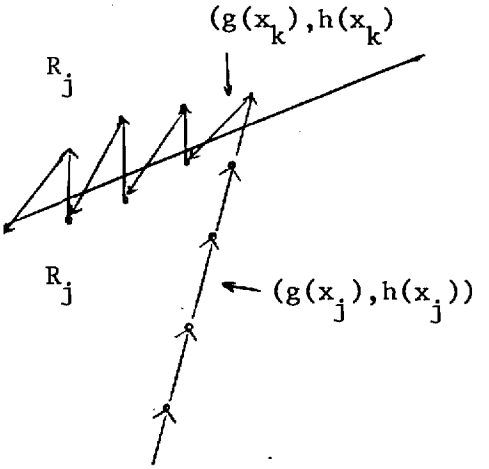
Moving in Discrete Steps

As long as we move along $(g(x_j,h(x_j))$ and stay in region R_j, θ is monotonically increasing. However, when we "step into" a new region, θ may or may not increase, since the gradient of the new region may point back into the region we just came from. At any rate, even if the last step did not incur any improvement, recalculating the minimal 1-tree at the new point will lead to a direction which improves θ, provided of course, we are in the interior of the new region. From this discussion it is clear that this type of discrete search is really a gradient search, rather than a subgradient search, because the likelihood of being exactly in the intersection of more than one region is small, and $(g(x_j),h(x_j))$ is indeed the gradient of θ provided that we are in the interior of R_j, i.e., when x_j is the unique minimal 1-tree. The two cases illustrated below are possible. In the first case, moving along the new gradient tends to take us into the interior of the new region, and hence we can move an appreciable distance across R_k, thus improving θ. The other case shows a situation where moving along

$(g(x_k),h(x_k))$ will take us back to R_j, i.e., $\langle (g(x_k),h(x_k)),(g(x_j),h(x_j))\rangle < 0$. This case corresponds to θ improving along the ridge $R_j \cap R_k$, and the search procedure described above will lead to zigzagging as shown below.



Moving Along New Gradient
Takes us "Away from R_j"

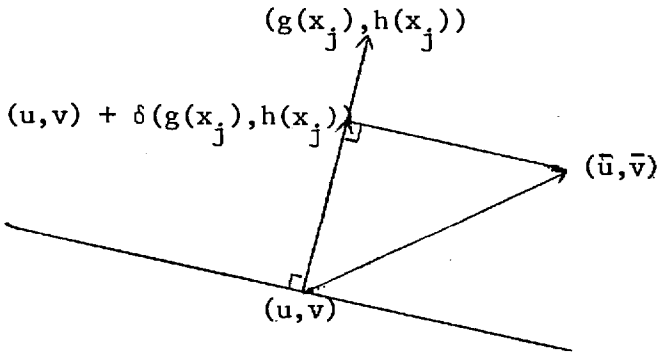


Moving Along New Gradient
Takes us Back to R_j "zigzagging"

It is also interesting to note, that even if moving along the direction $(g(x_j),h(x_j))$ (or any other subgradient of θ at (u,v)) starting from (u,v) does not improve the objective function, it will at least get us closer to the optimal point, provided that the stepsize is small enough. This fact has been shown by Held and Karp [11]. We will now illustrate this fact geometrically. Note that the half space $\{(\hat{u},\hat{v}): \langle(\hat{u},\hat{v}) - (u,v),(g(x_j,g(x_j))\rangle \geq 0\}$ contains the optimal point $(\bar{u},\bar{v})$. This is obvious since by concavity of θ we have

$$\theta(\bar{u},\bar{v}) \leq \theta(u,v) + \langle(\bar{u},\bar{v}) - (u,v),(g(x_j),h(x_j))\rangle$$

Since $\theta(\bar{u},\bar{v}) \geq \theta(u,v)$ then $\langle(\bar{u},\bar{v}) - (u,v),(g(x_j),h(x_j))\rangle \geq 0$. From this it is then clear that moving along the normal vector $(g(x_j),h(x_j))$ to the hyperplane $\{(\hat{u},\hat{v}): \langle(\hat{u},\hat{v}) - (u,v);(g(x_j),h(x_j))\rangle = 0\}$ starting from (u,v) leads closer to $(\bar{u},\bar{v})$. In fact $\|(\bar{u},\bar{v}) - (u,v) - \lambda(g(x_j),h(x_j))\|$ is a monotone decreasing function of λ for $\lambda \in [0,\delta]$ where δ is found such that the vectors $(g(x_j),h(x_j))$ and $(\bar{u},\bar{v}) - (u,v) - \delta(g(x_j),h(x_j))$ are orthogonal (see Figure below), i.e., $\delta = \dfrac{\langle(g(x_j),h(x_j)),(\bar{u},\bar{v}) - (u,v)\rangle}{\|(g(x_j),h(x_j))\|^2}$.



How Far to Go?

Even though the above analysis will not immediately lead to finding a suitable stepsize (since $(\bar{u},\bar{v})$ is not known at this stage), it at least establishes the fact that even if the objective function does not improve by moving along $(g(x_j),h(x_j))$ (if (u,v) is in the intersection of more than one region), we at least get closer to the optimal point, provided that the stepsize is small enough. As we get closer to the optimal point $(\bar{u},\bar{v})$, the vector $(\bar{u},\bar{v}) - (u,v)$ becomes small in norm and noting the form of δ it becomes clear that the stepsize should be small.

Zigzagging and Averaging of Directions

The case of zigzagging along the ridge $R_j \cap R_k$ may be time consuming. An averaging, or taking a suitable convex combination of the directions at any two successive iterations, will help reduce the zigzagging. This will in effect act as a projection of either of the gradient vectors on $R_j \cap R_k$. Calculating the projection vector along $R_j \cap R_k$ is easy, but the projection in the intersection of several regions may be tedious. Of course, if we are moving across the interior of a region R_j, then averaging of the gradient vector $(g(x_j),h(x_j))$ and itself will result in $(g(x_j),h(x_j))$, i.e., no harm is done!

Stepsize

Suppose we are at the point (u,v) with miminal 1-tree x_j. If (u,v) is in the interior of R_j, which is the case quite often, then moving along $(g(x_j),h(x_j))$ a small distance such that we are still in region R_j will improve the objective function. If we are very close to the boundary of R_j or on it and move along $(g(x_j),h(x_j))$ with a stepsize equal to λ, the objective function may or may not improve. Even if it does not improve, we get closer to the optimal point, provided that λ is small enough. At any rate, in order to implement the above procedure, we have to choose a suitable stepsize (or stepsizes). Normalizaton of the subgradient vector $(g(x_j,h(x_j))$ will be helpful so that the stepsize becomes more meaningful. According to our computational experience, we found that the stepsize does not play a major role in affecting the computational time for the traveling salesman problem, especially since we are seeking "close to optimal" solu-

tions of the dual problem rather than true optimal solutions. Discrepancy between suboptimal dual soltuions and the optimal dual solution, as well as any duality gap, will be compensated by the branch and bound scheme. We found that a value of $\lambda = 1$ is quite satisfactory. As a matter of fact, a value of λ anywhere from 1 to 5 was found satisfactory for all the problems we tried. A combination of different strategies can also be used, starting with a larger λ and then switching to a smaller λ as we get no improvement in the value of θ for several iterations.

Subgradient Optimization Algorithm

We will now describe a simple subgradient algorithm for maximizing θ. As in the previously discussed strategies we may not reach the true optimal of the dual function but rather get sufficiently close to it. Convergence of the procedure with sufficiently small λ is shown in [11]. We will use the a fixed value of λ (say $\lambda = 1$) in the following description with the understanding that different monotone values as the procedure continues can be used. At each iteration a minimal 1-tree is calculated. Different choices of the initial node can be made. Here n problems are initially solved with the initial dual variables $u_1 = v_1 = 0$, each time finding the minimal 1-tree with a different initial node. The initial node throughout the algorithm is the one with maximum minimal 1-tree.

Initialization Step

Choose (u_1, v_1), say $u_1 = v_1 = 0$, or u_1 and v_1 the dual variables of the corresponding assignment problem. Let $u^* = u_1$ and $v^* = v_1$. Let

$k = 1$, $\theta^* = -\infty$, $\gamma = 0$, and $\nu_{max} = T$. (T is the maximum number of iterations allowed without improving θ.) Choose λ and go to Step 1.

Step 1 (Finding $\theta(u_k, v_k)$)

Solve the following minimal 1-tree subproblem.

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{i=1}^{n} (c_{ij} + u_{ik} + v_{jk}) x_{ij}$$

Subject to: $x \in X$

where $u_k = (u_{1k}, \ldots, u_{nk})$ and $v_k = (v_{1k}, \ldots, v_{nk})$

Let x_k be an optimal 1-tree. If x_k is a directed loop, then stop, the optimal primal and dual solutions have been found. Otherwise let $\theta(u_k, v_k) = \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{i=1}^{n} (c_{ij} + u_{ik} + v_{jk}) x_{ij} - \sum_{i=1}^{n} u_{ik} - \sum_{j=1}^{n} v_{jk}$.

If $\theta(u_k, v_k) > \theta^*$, then let $\nu = 0$, replace θ^* by $\theta(u_k, v_k)$, replace u^* by u_k and v^* by v_k, and go to Step 2. If on the other hand, $\theta(u_k, v_k) \leq \theta^*$, replace ν by $\nu + 1$. If $\nu = \nu_{max}$, then stop with θ^* and (u^*, v^*). If $\nu < \nu_{max}$, go to Step 2.

Step 2 (Update dual variables)

Let $u_{k+1} = u_k + \lambda g(x_k)$ and $v_{k+1} = v_k + \lambda h(x_k)$. Replace k by $k+1$ and repeat Step 1.

Finding the Optimal Primal Solution: Branch and Bound

So far we have described various strategies for finding the optimal (or near optimal if ascent search is not fully completed) dual solution. In the process, the optimal loop of the traveling salesman problem may be found. If this were not the case (either a duality gap exists or

dual is not solved exactly), a branch and bound scheme for finding the optimal directed loop will be described. It is important to note that even in this case, optimizing the dual function θ played the following significant roles:

1. Provided a "tight" lower bound on the optimal traveling salesman problem and hence making the notion of "verified" suboptimal solutions attractive.
2. Rather than using the cost c_{ij} of link ij in the branch and bound scheme, c_{ij} is replaced by $c_{ij} + u_i + v_j$, where u and v are the optimal dual variables, thus resulting in a "well-conditioned" cost matrix. The computational effort for finding the optimal tour, using the new cost matrix $c_{ij} + u_i + v_j$ is considerably smaller than the effort to be expended, had the original cost matrix been used in a branch and bound scheme. This is due to the fact that the new cost matrix $(c_{ij} + u_i + v_j)$ is "conditioned" in the sense that the optimal 1-tree with the above matrix is "close" to the optimal traveling salesman tour.

Consider now the following traveling salesman problem:

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} (c_{ij} + u_i + v_j)\, x_{ij}$$

$$\text{Subject to:} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} = 1 \qquad \text{for } i = 1, 2, \ldots, n$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1 \qquad \text{for } j = 1, 2, \ldots, n$$

No Subtours

Note that an optimal tour obtained by the above problem is an optimal solution of the original traveling salesman problem. A branch and bound procedure will be used to find the optimal tour.

General Framework

A branch and bound procedure will be used to obtain the optimal tour. A list of distinct nodes $i_1, i_2, i_3, \ldots, i_k$ and corresponding links $(i_1,i_2), \ldots, (i_{k-1},i_k)$ form a partial solution. Since the nodes are distinct, no subtours are a part of the partial solution. If $k = n$, then a complete tour is obtained by adding the link (i_k,i_1). Suppose that we are given a partial solution, and attempt to find a feasible completion, i.e., a tour. A lower bound on the cost of completing this partial solution is computed, say LB. If $LB \geq C^*$, where C^* is the cost of the best tour known so far, then the partial solution is fathomed, since there is no way to find a better solution by completing it. In this case the partial solution is abandoned. If $LB < C^*$ then an additional link (i_k,i_{k+1}) is added to the partial tour where i_{k+1} is not equal to i_j for $j = 1, 2, \ldots, k$. The method of choosing i_{k+1}, referred to as the branching node, and the method of calculating the lower bound, completely define the search.

Calculation of the Lower Bound

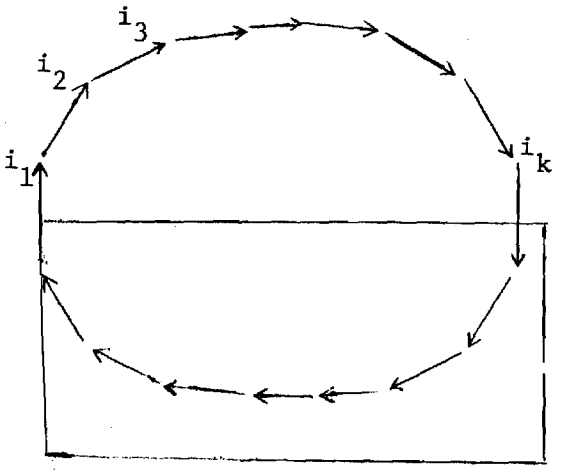
Suppose that $i_1, i_2, \ldots, i_k$ form a partial solution. If (i_j,i_{j+1}) for $j = 1, 2, \ldots, k-1$ are to be a part of a tour, then the remaining nodes must themselves form a "string", which is connected to nodes i_1 and i_k. Since the "string" is a tree spanning the remaining nodes, then its cost must be greater or equal to the minimal cost of

all trees spanning these nodes. Also, the cost of connecting the partial solution to this open loop is greater or equal to the sum of the following:

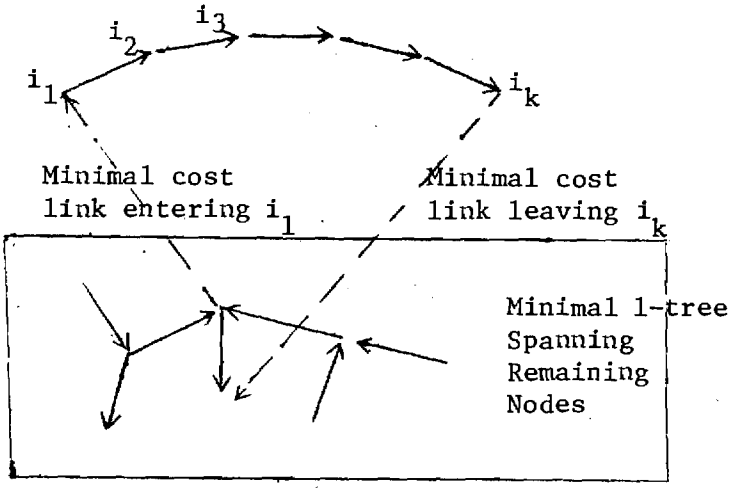
1. Minimal cost of (available) links leaving node i_k
2. Minimal cost of (available) links entering node i_1

We use the word <u>available</u>, since links (i_j, i_1) for $j = 2, \ldots, k$ and (i_k, i_j) for $j = 1, 2, \ldots, k-1$ are not available, because their use will form a subtour. Also, as the search continues, other links will be prohibited from usage.

We show below a feasible completion of a partial solution, and also a method for calculating a lower bound on the cost of all completions of this partial solution.



A Feasible Completion
of the Partial Solution



Calculation of Lower Bound on All
Completions of Partial Solution

From this it is clear that a lower bound LB on the cost of the partial solution, plus the cost of its completion, is given by:

$LB = C_1 + C_2 + C_3$, where

C_1 : the actual cost of the partial solution

$$C_1 = \sum_{j=1}^{k-1} (c_{i_j i_{j+1}} + u_{i_j} + v_{i_{j+1}})$$

C_2 : the cost of the minimal tree spanning the remaining nodes, where the cost of link ij is $c_{ij} + u_i + v_j$. There are efficient methods for calculating C_2 (see [ ]).

C_3 : Minimal cost of "connecting" the partial solution to the spanning tree.

$$C_3 = \underset{j \in A}{\text{Minimum}} (c_{i_k j} + u_{i_k} + v_j) + \underset{j \in B}{\text{Minimum}} (c_{j i_1} + u_j + v_{i_1})$$

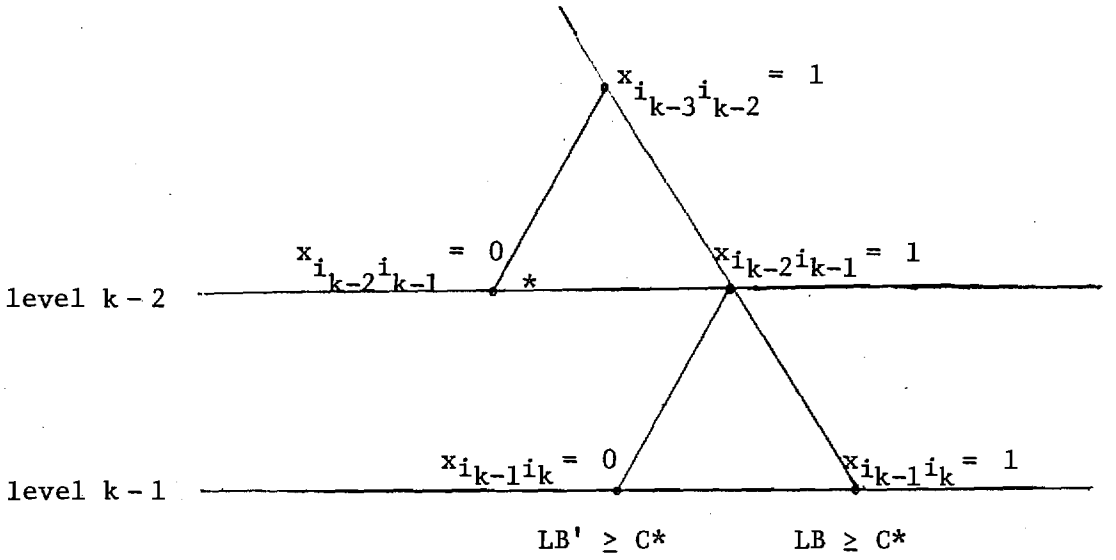
where A is the set of nodes which i_k is eligible to enter, and B is the set of nodes which are eligible to enter node i_1.

Continuation of the Search

1. Fathoming (Backward Move)

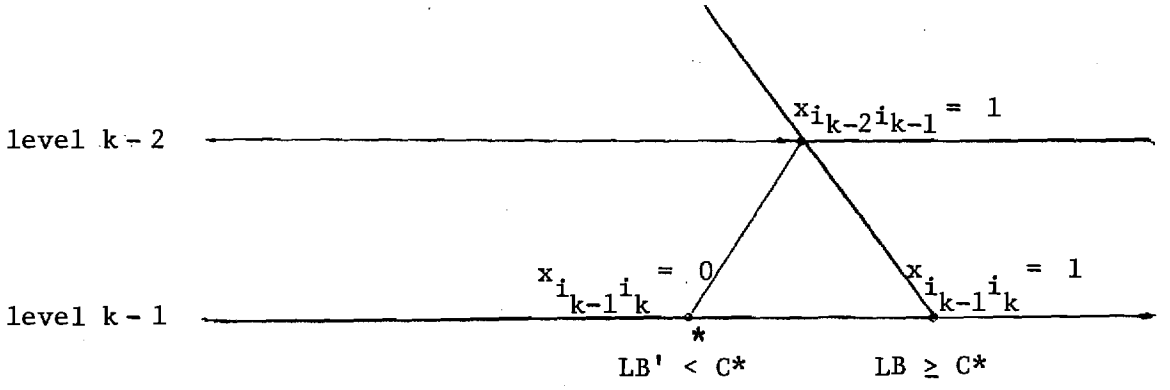
Suppose that we have a partial tour $i_1, i_2, \ldots, i_k$; i.e., $x_{i_1 i_2} = x_{i_2 i_3} = \ldots, = x_{i_{k-1} i_k} = 1$. The level of the search tree is called k - 1. The lower bound LB on the cost of the partial solution plus its completion is calculated as shown above. If $LB \geq C^*$, where C^* is the current best known cost of a complete tour, then pursuing this partial solution is not worthwhile, and the partial solution is said to be fathomed. The last move from i_{k-1} to i_k is prohibited, i.e., $x_{i_{k-1} i_k} = 0$. A new bound LB' on completing the cost of the partial tour $i_1, \ldots, i_{k-1}$ with the additional restriction that $x_{i_{k-1} i_k} = 0$ is computed. This is done in the same manner as before except of

course that i_k now belongs to the remaining nodes and the link $i_{k-1}i_k$ cannot be used. If $LB' \geq C^*$, then the partial solution $i_1, i_2, \ldots, i_{k-1}$, with the added restriction $x_{i_{k-1}i_k} = 0$ can lead to no improved solutions. This means that all possibilities at level $k-1$ are exhausted. This case will be called a strong fathoming, and the level of the tree is reduced by 1 unit. In this case $x_{i_{k-1}i_k} = 0$ is not prohibited any more, whereas $x_{i_{k-2}i_{k-1}}$ is forced to be zero. The process is then repeated. If on the other hand $LB' < C^*$, which will be called weak fathoming, then we seek to find a node j such that $x_{i_{k-1}j} = 1$. This step will be discussed in more detail in the forward move below. The cases of strong and weak fathoming are illustrated below.



Strong Fathoming

* Continue at level k - 2 by trying to go from node i_{k-2} to a node $j \neq i_{k-1}$



Weak Fathoming

* Continue at level k - 1 by trying to go from node i_{k-1} to node $j \neq i_k$.

2. Forward Move

If the lower bound is less than C*, then it may be possible to find a completion of the partial solution with an objective less than C*. Suppose that the current level of the tree is $k-1$, where at level $k-1$ we have $x_{i_{k-1}i_k} = 1$. Now we seek a node $i_{k+1} \neq i_j$ for $j = 1, 2, \ldots, k-1$ and the move $i_k i_{k+1}$ is not prohibited from previous fathoming (if such a node i_{k+1} is not found then the partial solution is fathomed and the level of the tree is reduced). Many rules (branching rule) for choosing i_{k+1} are possible. One of the simplest (and most effective for the traveling salesman problem) is to choose i_{k+1} which minimizes $c_{i_k j} + u_{i_k} + v_j$ for all available j. Another possible method is to calculate a lower bound LB_j for each possible node j, and then choose the one with the least possible lower bound to be i_{k+1}. If the choice of i_{k+1} is determined by this rule, then we automatically have the lower bound needed for deciding whether to fathom, and we also may prohibit some of the moves $i_k j$ if $LB_j \geq C^*$.

At any rate, i_{k+1} is chosen, the level of the tree is increased by 1, and a lower bound (if the first rule was used) is calculated, and the process is repeated. Needless to say, if the level is $n-1$, then i_n is joined with i_1 and the cost of the link $i_n i_1$ is added to the cost of the partial tour. If the total cost is less than C* then C* is replaced by the total cost, and the tour is stored as the best tour obtained so far.

3. Termination

We have described forward and backward progress of the search. If the level of the search ever reaches value zero, then we stop. This would mean that we are currently at level 1, and are trying to backtrack. This means that all possible completions of $x_{i_1 i_2} = 1$ and $x_{i_1 i_2} = 0$ have been enumerated, i.e., all the search space has been exhausted. The stored loop and C* give the optimal solution.

Summary of the Algorithm

Initialization Step

Let $P(i) = \emptyset$ for all $i = 1, 2, \ldots, n$ and let $C^* = \infty$. Choose node i_1 (according to the procedure described earlier) as the initial node in the maximization of θ. Let $k = 0$ and go to Step 1.

Step 1 (Forward Move)

If $k = n-1$, replace C_1 by $C_1 + (c_{i_n i_1} + u_{i_n} + v_{i_1})$. If $C_1 < C^*$, then replace C* by C_1, store the complete solution $i_1, \ldots, i_n, i_1$ and go to Step 2. If $C_1 \geq C^*$, go to Step 2. If $k < n-1$ calculate a lower bound LB on all completions of the partial solution at hand, where $LB = C_1 + C_2 + C_3$, where C_1, C_2, and C_3 are calculated as described above, $A = P(i_k) \cup \{i_1, \ldots, i_{k-1}, i_k\}$ and $B = \{i_1, i_2, \ldots, i_k\}$. If $LB \geq C^*$, go to Step 2. Otherwise pick $i_{k+1} \notin A$ such that $c_{i_k i_{k+1}} + u_{i_k} + v_{i_{k+1}} = \underset{j \notin A}{\text{Minimum}} (c_{i_k j} + u_{i_k} + v_j)$. Replace k by $k+1$ and repeat Step 1.

Step 2 (Fathom)

Replace $P(i_{k-1})$ by $P(i_{k-1}) \cup \{i_k\}$. Calculate a lower bound LB on all completions of the partial solution $i_1, i_2, \ldots, i_{k-2}, i_{k-1}$. If $LB \geq C^*$,

go to Step 3. Otherwise choose a node $j \notin P(i_{k-1})$ such that $c_{i_{k-1}j} + u_{i_{k-1}} + v_j$ = Minimum $\{ c_{i_{k-1}i} + u_{i_{k-1}} + v_i \mid i \notin P(i_{k-1}), i \neq i_1, \ldots, i \neq i_{k-1} \}$. Denote the new i_k by j. This forms a new partial solution. Go to Step 1.

Step 3 (Strong Fathoming)

i_{k-1} is placed in $P(i_{k-2})$. $P(i_{k-1})$ is replaced by $\emptyset$, and k is replaced by k - 1. If k = 0, go to Step 4, otherwise go to Step 2.

Step 4 (Termination)

Search has been completed. Optimal cost is C* and its corresponding tour is $i_1, i_2, \ldots, i_n, i_1$. Stop.

Optimal and Suboptimal Solutions via "Controlled" Fathoming

Rather than attempting to fathom a partial solution when the lower bound LB on its completion is greater than or equal to the best known C*, we may fathom if LB is "Close" to C*. This will enable us to fathom quickly and hence reduce the portion of the search tree explicitly enumerated. This procedure can be controlled either to obtain suboptimal solutions, with any desired degree of accuracy, or to obtain optimal solutions. In both cases the computational effort will be significantly reduced. The following two methods are proposed for implementing the notion of controlled fathoming.

Method 1

Recall that a partial solution is fathomed, if the lower bound on all completions is at least as big as C*, the best known objective. Suppose that a partial solution is fathomed if LB $\geq \alpha C^*$, where $\alpha \in (0,1]$. In

this case, the partial solution is abandoned if there is no hope that it leads to an objective which is better than αC^*. To illustrate, suppose that C^* is 1000, and we choose $\alpha = 0.95$. Then we fathom if $LB \geq 950$. This means that if the bound on completions of the partial solution cannot lead to an improvement of 50 units over what we already have, then it does not pay off to pursue this partial solution. The objective of this simple strategy is clear. We want to fathom, i.e., abandon partial solutions, quickly, even if they lead to a slight improvement. As the reported computational experience shows, this simple strategy speeds the computational effort considerably. Of course, as a new C^* is found, then we fathom whenever the bound is greater or equal to α times the new C^*. The procedure continues until we cannot find a feasible solution with an objective less than αC^*. So we have a feasible tour with objective C^*, coupled with the statement that the optimal objective is greater than or equal to αC^*. Needless to say, if αC^* is less than the overall lower bound, which is Max $\theta(u,v)$, then there is no need to complete the search, since we know that there are no solutions with objective less than αC^*. It is interesting to keep in mind that when we try to find a solution with objective less than αC^*, the computational effort will either lead to such a solution or conclude that no such solution exists. In the first case, we are able to improve the current best solution, and in the second case we are able to "tighten" the overall lower bound on the problem. In other words, a valuable information is obtained whether we are able to find a solution with objective less than αC^* or not.

Choice of α

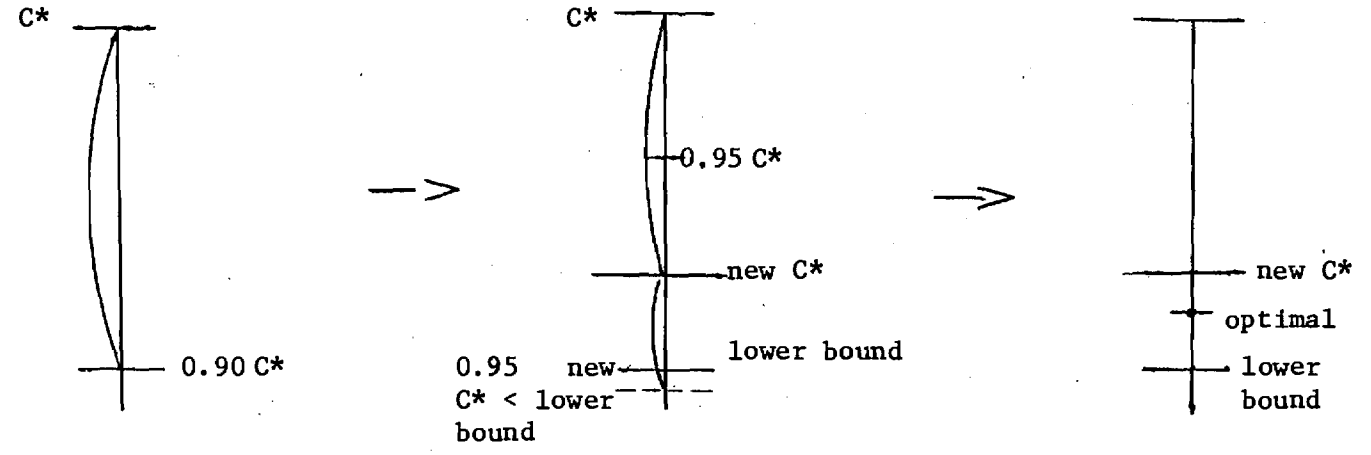
Of course, if α is small, then fathoming will speed up considerably, resulting in a small computational effort. But on the other hand, the quality of the best obtained feasible solution is not guaranteed. To illustrate, suppose that at termination C* is 1000 and suppose that $\alpha = 0.60$. All we can say in this case is that we have a feasible solution with objective 1000, and the true optimal has an obejctive greater or equal to 600. This, of course, is not satisfactory. Suppose, however, that we used $\alpha = 0.95$. Then if at termination C* is 1000, we can say that the optimal is greater than or equal to 950. This statement is indeed much stronger than the previous statement.

From this discussion, it is clear that there is a tradeoff between the computational effort and the value of α. We recommend values of $\alpha \geq 0.93$ depending on the accuracy required.

Exact Solution By Stepped Fathoming

A slight modification of the above scheme can be used to find the exact optimal solution. Suppose that an initial α_1 is used, say $\alpha_1 = 0.90$. Eventually we achieve a final C* and cannot find a feasible solution with objective less than $0.90\,C^*$ i.e., $0.9\,C^*$ is a lower bound on all solutions. We either stop here, or else switch to $\alpha_2 \geq \alpha_1$, say $\alpha_2 = 0.95$. We start the search from the stored best solution corresponding to C*, which also includes all the prohibited visitations. We are either able to find a feasible solution with objective less than $0.95\,C^*$ (none less than $0.9\,C^*$ exist), and repeat the process, or else conclude that none exist. We may then switch to 0.97 fathoming and eventually 100% fathom-

ing. This process is shown below and proved very efficient from a computational point of view.



no solution with objective < 0.9 C* exists

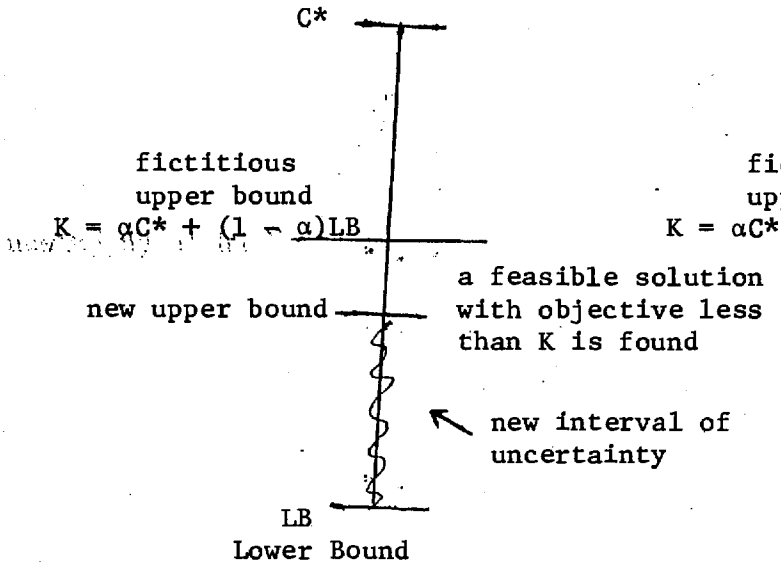
A solution with objective < 0.95 C* found

100% fathoming

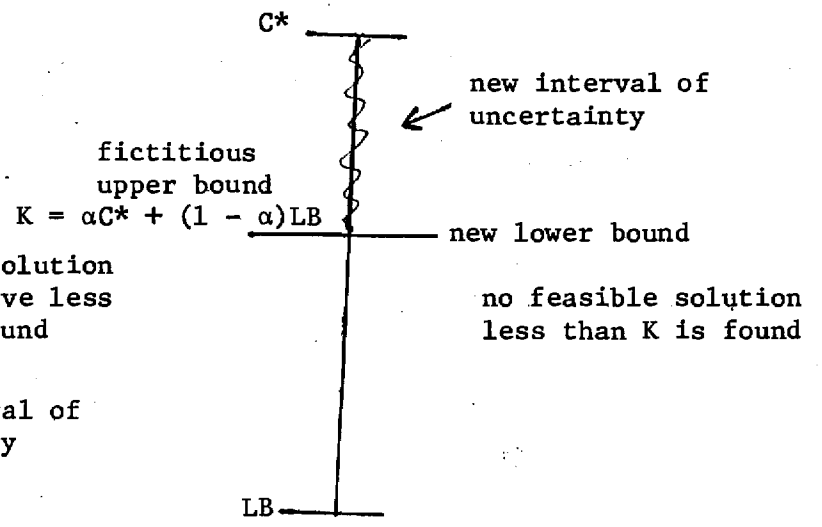
Method 2

This method fathoms whenever the lower bound LB is greater than or equal to K, where K is related to, but not necessarily equal to C* ($K \leq C^*$). Again we have the same interpretation as before. A partial solution is pursued if it guarantees a nontrivial improvement over what we got. We will describe a procedure to find a suitable K. At each stage of the algorithm, we have an upper bound (optimal $\leq$ upper bound), namely C*. A lower bound on the overall problem, namely, Max $\theta(u,v)$ is also known. At each stage of the algorithm we have a feasible solution with objective C*, and we know that the optimal must be less than or equal to C*. Rather than fathoming on C*, suppose we fathom on $K = \alpha C^* + (1 - \alpha)LB$, where $\alpha \in (0,1]$. Since LB < C*, then $K = \alpha C^* + (1 - \alpha)LB \leq C^*$. Two cases are possible. In the first case, we will be able to find a tour with objective less than K. The objective of this new solution becomes the new upper bound C* and the process is repeated. In the second case, we will not be able to find such

a solution. This automatically implies that there are no solutions with objective less than K, and hence K itself is the new lower bound. The process is repeated. From this we keep narrowing the gap between the lower and upper bounds, either by lowering the upper bound (finding an improved feasible assignment) or by raising the lower bound (by finding that there is no feasible solution with objective lower than the current lower bound). When the difference between the lower bound, and the current best objective value (upper bound) is smaller than a prescribed tolerance, we either stop, or try to find solutions with objective less than C* itself, i.e., we switch to $\alpha = 1$. Of course, if the tolerance is zero, or if we switch the fathoming to C* itself, then we will find the true optimal solution.



Upper bound reduced



Lower bound raised

Choice of α

α is any number in the interval (0,1]. Of course, if α is close to 1, then we are in effect fathoming on a number very close to C*, and the search will not speed considerably. On the other hand, if α is close to zero, then we only fathom if we obtain a feasible solution very close to the overall lower bound. In this case, fathoming will be fast, but it is likely not to obtain feasible solutions less than K. The value $\alpha = 0.5$ is recommended so that the distance of uncertainty is halfed at each time. When the lower and upper bounds are close to each other, we switch to $\alpha = 1$.

References

1. Bazaraa, M. S. and J. J. Goode, Lagrangian Duality In Mathematical Programming: A Unified Approach. (In preparation), Georgia Institute of Technology, Atlanta, Georgia.

2. Balas, E., An Infeasibility-Pricing Decomposition For Linear Programs, Operations Research, Vol. 14, 1966, pp. 847-873.

3. Dantzig, G. B. and P. Wolfe, The Decomposition Algorithm For Linear Programming, Econometrica, Vol. 29, No. 4, 1961.

4. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, 1963.

5. Dijkstra, E. W., A Note On Two Problems In Connexion with Graphs, Numerische Mathematik, Vol. 1, 1959, pp. 269-271.

6. Fisher, M. L., W. D. Northup, and J. F. Shapiro, Using Duality To Solve Discrete Optimization Problems: Theory and Computational Experience, Working paper OR 030-74, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts, January, 1974.

7. Geoffrion, A. M., Lagrangian Relaxation and Its Uses In Integer Programming, Working paper No. 195, Western Management Science Institute, December, 1972.

8. Geoffrion, A. M., Elements of Large Scale Mathematical Programming, Management Science, Vol. 16, No. 11, 1970, pp. 652-691.

9. Grinold, R. C., Steepest Ascent For Large Scale Linear Programs, SIAM Review, Vol. 14, 1972, pp. 447-464.

10. Held, M. and R. M. Karp, The Traveling Salesman Problem and Minimal Spanning Trees, Operations Research, Vol. 18, No. 6, 1970, pp. 1138-1162.

11. Held, M. and R. M Karp, The Traveling Salesman Problem and Minimum Spanning Trees: Part II, Mathematical Programming, Vol. 1, 1971, pp. 6-25.

12. Held, M., P. Wolfe, and H. D. Crowder, Validation of Subgradient Optimization, I B M Report RC 4462, August, 1973.

13. Kruskal, J. B., On The Shortest Spanning Subtree Of A Graph And The Traveling Salesman Problem, Proc. Am. Math. Soc., Vol. 1, 1956, pp. 48-50.

14. Lasdon, L. S., Optimization Theory For Large Systems, The Macmillan Company, 1970.

15. Rockafellar, R. T., Convex Analysis, Princeton University Press, 1970.

16. Stoer, J. and C. Witzgall, Convexity And Optimization In Finite Dimensions I, Springer Verlag, 1970.

17. Zangwill, W. I., Nonlinear Programming: A Unified Approach, Prentice-Hall, 1969.

Personnel

Principal Investigator

Dr. Mokhtar S. Bazaraa
Associate Professor
Industrial and Systems Engineering

Organization and Coordination of the project
Developing Quadratic Assignment and Traveling Salesman Algorithms
Review of Programming Development

Graduate Research Assistants

1. Virginia Ruth Khege
   Computer Program of Initial Solutions of Quadratic Assignment and Traveling Salesman Problems

2. Peter H. Geddes
   Applications of Quadratic Set Covering and Assignment Problems

3. Sinnan S. Tumer
   Programming of Quadratic Assignment and Set Covering Algorithms

List of Papers Related to Project

1. M. S. Bazaraa and A. N. Elshafei, The Concept of Stepped Fathoming In Tree Search Algorithms, submitted to Management Science.

2. M. S. Bazaraa and A. N. Elshafei, Heuristic and Exact Procedures of the Quadratic Assignemnt Problem, In preparation.

3. M. S. Bazaraa and J. J. Goode, The Traveling Salesman Problem: A Duality Approach, In preparation.

4. M. S. Bazaraa, Layout Design As A Quadratic Set Covering Problem, In preparation.

E-24-618
Final Tech.

THE QUADRATIC SET COVERING (ASSIGNMENT) PROBLEM: APPLICATIONS AND COMPUTATION

PRESENTED TO THE
NATIONAL SCIENCE FOUNDATION
UNDER NSF GRANT
GK - 38337

PRINCIPAL INVESTIGATOR
MOKHTAR S. BAZARAA

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY

FEBRUARY 1975

THE QUADRATIC SET COVERING (ASSIGNMENT) PROBLEM: APPLICATIONS AND COMPUTATION

Presented to The National Science Foundation

under NSF Grant GK-38337

Principal Investigator

Mokhtar S. Bazaraa

School of Industrial and Systems Engineering

Georgia Institute of Technology

February 1975

TABLE OF CONTENTS

Page

I. INTRODUCTION

I.1 Summary and Highlights of the Report

Linear set covering and set partitioning problems have attracted the attention of many researchers. This is due to the simple formulation of these problems and their wide range of applications.

This research concerns itself with the quadratic set covering problem, the quadratic assignment problem, and other related problems. The significance of these problems is reflected by the flexibility of the quadratic objective, and its ability to deal with first order interaction between the variables, which leads to a large number of applications ranging from location of facilities, computerized layout design, sequencing of jobs on a machine, political districting, to selection of projects.

The following are the main results of the research.

--A unified presentation of the quadratic set covering, the quadratic assignment, and related problems. Applications of these problems in the areas of location, computerized layout, routing, project selection, sequencing, and scheduling are unified.

--Development of an algorithm for finding optimal and sub-optimal solutions of quadratic set covering problems, in the general context of layout design.

--Development of exact and heuristic algorithms for solving the quadratic assignment problem.

--Development of an exact algorithm for solving symmetric and nonsymmetric traveling salesman problems via branch and bound.

I. INTRODUCTION

I.1 Summary and Highlights of the Report

Linear set covering and set partitioning problems have attracted the attention of many researchers. This is due to the simple formulation of these problems and their wide range of applications.

This research concerns itself with the quadratic set covering problem, the quadratic assignment problem, and other related problems. The significance of these problems is reflected by the flexibility of the quadratic objective, and its ability to deal with first order interaction between the variables, which leads to a large number of applications ranging from location of facilities, computerized layout design, sequencing of jobs on a machine, political districting, to selection of projects.

The following are the main results of the research.

--A unified presentation of the quadratic set covering, the quadratic assignment, and related problems. Applications of these problems in the areas of location, computerized layout, routing, project selection, sequencing, and scheduling are unified.

--Development of an algorithm for finding optimal and sub-optimal solutions of quadratic set covering problems, in the general context of layout design.

--Development of exact and heuristic algorithms for solving the quadratic assignment problem.

--Development of an exact algorithm for solving symmetric and nonsymmetric traveling salesman problems via branch and bound.

I.2 Organization of the Report

In Section II of the report, a classification of the quadratic set covering and related problems is presented. Different known and new applications of these problems are also presented in Section II. The remaining sections are devoted to the development of solution procedures for this class of problems. In Section III a branch and bound algorithm for solving quadratic set covering problems which arise in the context of layout design is developed. Both branch and bound and heuristic algorithms for solving quadratic assignment problems are developed in Section IV. Finally, in Section V a special procedure based on the concept of duality is developed for solving symmetric and nonsymmetric traveling salesman problems. In each case, random as well as standard problems in the literature are solved to illustrate computational effectiveness of the procedures developed. Appendices A, B, and C reproduce the algorithms of this study.

II. THE QUADRATIC SET COVERING PROBLEM, RELATED PROBLEMS, AND APPLICATIONS

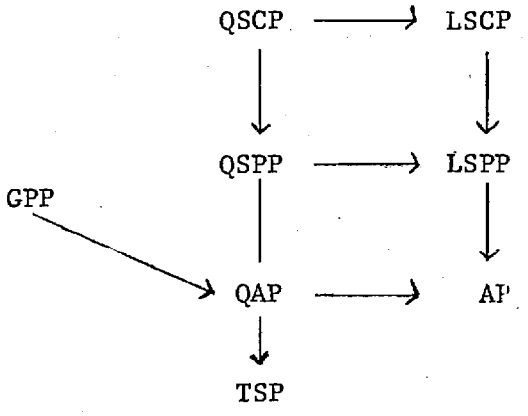
In this section we will describe the quadratic set covering problem and other related problems, and then give a number of examples illustrating the applications of these problems.

II.1 Mathematical Descriptions

We shall consider the following problems:

quadratic set covering problem	QSCP
quadratic set partitioning problem	QSPP
linear set covering problem	LSCP
linear set partitioning problem	LSPP
generalized placement problem	GPP
quadratic assignment problem	QAP
assignment problem	AP
traveling salesman problem	TSP

All of the problems, except the GPP which is new, are well known. As the problems are precisely described below, it will be helpful to keep in mind the following diagram showing the relationships of the problems as special cases of the others in the formulations we give.



Let x be an n x 1 binary vector, x' the transpose of x, A an m x n matrix of zeros and ones, C an n x n matrix, and e_m an m x 1 vector of ones. The quadratic set covering problem (QSCP) is

$$\text{Minimize} \quad x'Cx$$

$$\text{such that } Ax \geq e_m$$

$$x \text{ is a binary vector}$$

and the quadratic set partitioning problem (QSPP) is

$$\text{Minimize} \quad x'Cx$$

$$\text{such that } Ax = e_m$$

$$x \text{ is a binary vector}$$

If we let I_n be the n x n identity matrix, then by choosing A to be the $2n \times n^2$ matrix

$$A = \begin{bmatrix} e' & & & \\ & e' & & \\ & & \ddots & \\ & & & e' \\ I_n & I_n & \cdots & I_n \end{bmatrix}$$

we obtain the quadratic assignment problem (QAP) for the case of assigning n objects to n locations. We may always assume, without loss of generality, that the number of objects to be assigned is equal to the number of locations, for if this is not the case, then we may add dummy objects which make them equal but have no effect on the final solution. If the objective is of the form $d'x + x'Cx$, the problem is usually called a placement problem. The additional linear term represents no

generalization, however, since the quadratic objective can also handle the linear terms by placing the entries d_j's along the diagonal of C. Therefore, we give no special considerations to placement problems as they are treated within the general class of quadratic set covering and set partitioning problems. By replacing the quadratic objective functions of the QSCP, QSPP, and QAP with a linear function, we obtain the linear set covering problem (LSCP), linear set partitioning problem (LSPP), and assignment problem (AP) respectively. The traveling salesman problem (TSP), the problem of a salesman who must visit n cities, each exactly once, and return home while minimizing the total distance of his trip will be formulated later as a special case of the QAP. The generalized placement problem (GPP) allows one to require that certain sets of objects must be assigned to certain sets of locations, or certain sets of locations must be filled by objects from certain sets of objects (Geddes [13]). As will be seen in the applications, these are very realistic restrictions, and when the GPP is formulated in terms of 0-1 variables the complications are actually helpful in obtaining the solution more efficiently. Before describing the GPP we need to introduce some notation. For a finite set S let $|S|$ = the number of elements of S and $P(S) = \{X|X \subset S\}$ = the power set of S. Suppose we are given two finite sets A and B with $|A| = |B|$. (Since A is thought of as a set of elements to be assigned to locations B, we must have $|A| \leq |B|$. There is no loss generality in assuming $|A| = |B|$, since "dummy" elements may always be added to A which do not influence the objective function.) $\Phi = \{\psi|\psi: A \to B, \psi \text{ 1-1 map}\}$. Assume that fixed assignment costs of elements of A to B, quantities shipped, and costs of shipping have been taken into consideration to produce a quadratic objective function, as in the QAP, and let c_ψ be the cost associated with $\psi \in \Phi$. Let

$R \subseteq P(A) \times P(B)$. We say that an element $\psi \in \Phi$ satisfies R, written ψSR, provided:

$$\text{for each } (\hat{A},\hat{B}) \in R, \quad \psi[\hat{A}] \subset \hat{B} \text{ if } |\hat{A}| < |\hat{B}|$$

$$\psi^{-1}[\hat{B}] \subset \hat{A} \text{ if } |\hat{A}| > |\hat{B}|$$

$$\psi[\hat{A}] = \hat{B} \text{ if } |\hat{A}| = |\hat{B}|$$

Here $\psi[\hat{A}]$ denotes the image of the set $\hat{A}$ under the map ψ. The generalized placement problem (GPP) is

$$\begin{aligned} &\text{minimize} && c_\psi \\ &\text{subject to} && \psi \in \Phi \\ &\text{and} && \psi SR \end{aligned}$$

where A, B, R, and the costs c_ψ are given. Note that if $R = \{(A,B)\}$ we simply have the QAP. Other cases, such as when R is of the form $R = \{(\{a_{i_1}\}, \{b_{i_1}\}), \ldots (\{a_{i_s}\}, \{b_{i_s}\})\}$, $s < |A|$, will be considered in sections II.2 and II.3, where examples are given for location problems and the TSP. A procedure is also given for writing the GPP in terms of 0-1 variables.

II.2 The Location Problems

Because of their similar mathematical structure as quadratic assignment problems, as well as analogous physical and economic interpretations, we give four applications under the general classification of location problems. These are

a. facilities location
b. building layout
c. control panel layout
d. wiring problem

Each application will be discussed in detail individually, but first the location problem in general, including properties and extensions common to all of the applications, is described. We suppose that n objects a_i $(i = 1,2,\ldots,n)$ are to be assigned to n locations b_k

$(k = 1, 2, \ldots, n)$ in such a way that each object is assigned to one and only one location. As was mentioned earlier, it is easily seen that if we have m objects and n locations with $m < n$, we may supply $n - m$ dummy objects so that there is no loss of generality in assuming the same number of objects and locations. There may or may not be a fixed cost f_{ik} associated with assigning object a_i to location b_k. There are given a fixed number of "units" to be "shipped" from object a_i to object a_j, say u_{ij}, and in general $u_{ij} \neq u_{ji}$. These units may be interpreted as trips made by people in the building layout problems, as we shall see, or as numbers of wire connections in the wiring problem. Shipping one unit from location b_k to b_ℓ incurs a cost $d_{k\ell}$ (usually related to distance). Define n^2 variables x_{ik} $(i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, n)$ by

$$x_{ik} = \begin{cases} 1 & \text{if object } a_i \text{ is placed in location } b_k \\ 0 & \text{otherwise} \end{cases}$$

and let $x' = (x_{1,1}\ x_{1,2} \cdots x_{1,n} \cdots x_{n,1}\ x_{n,2} \cdots x_{n,n})$.
For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$, define n^2 matrices of order $n \times n$

$$C_{ij} = \left[C \begin{smallmatrix} ij \\ rs \end{smallmatrix} \right] \qquad (r = 1, 2, \ldots, n;\ s = 1, 2, \ldots, n)$$

where, if $i = j$, then $C \begin{smallmatrix} ij \\ rs \end{smallmatrix} = \begin{cases} f_{ir} & \text{if } r = s \\ 0 & \text{if } r \neq s \end{cases}$

$$\text{if } i \neq j, \text{ then } C \begin{smallmatrix} ij \\ rs \end{smallmatrix} = \begin{cases} 0 & \text{if } r = s \\ u_{ij} d_{rs} & \text{if } r \neq s \end{cases}$$

$C \begin{smallmatrix} ij \\ rs \end{smallmatrix}$ is therefore interpreted as the cost of shipping u_{ij} units from object a_i to object a_j, if they were located in b_r and b_s respectively. Let C be the partitioned matrix

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & & & \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}$$

Let e_n be a n x 1 vector of ones and I_n be the n x n identity matrix.

Let H be the 2n x n^2 assignment matrix

$$H = \begin{bmatrix} e_n' & & & \\ & e_n' & & \\ & & \ddots & \\ & & & e_n' \\ I_n & I_n & \cdots & I_n \end{bmatrix}.$$

The location problem (quadratic assignment problem) is to

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_{2n}$$

$$x \text{ binary}$$

It should be noted that a large number of elements of the C matrix were set equal to zero since they correspond to assignments not permitted by $Hx = e_{2n}$. The first n rows of equations represented by $Hx = e_{2n}$ require that each object is assigned to one location, and the remaining n equations require that each location is assigned to one object; that is to say we have a one-to-one correspondence between objects and locations. Now, for example, the object a_1 cannot be assigned to locations b_1 and b_3 simultaneously, so the first row third column entry of C_{11}, C_{13}^{11}, was set equal to zero. Actually, any number could be placed

in this entry, since it would never contribute to the value of the objective function $x'Cx$ for any feasible x.

There are two extensions of the location problem which fall into the case of the GPP. In one of these we may require a few selected objects to be assigned to particular locations. For example, we may be given a collection of factories and warehouses already built and wish to add some new ones. A slightly different problem occurs when certain sets of objects must be assigned to certain sets of locations, or certain sets of locations must be filled by objects from given sets. In the second instance, we still have some freedom to locate the object. An example is that the tuning knob on a radio has to be, say, on the outside in front and not buried in the middle of the set, but there are several locations available. As another example, consider the problem of assigning ten plants of various sizes, some of which require a water supply, to sites of different sizes, some of which have water supplies. This type of constraint may also be used to keep certain objects either close together or far apart. In a warehouse certain chemicals cannot be stored together for safety reasons such as fire and explosion. Certain areas in a warehouse may be unsuitable for storing some materials because of temperature or humidity, while some materials may be stored in any area of the warehouse. We may wish to require certain rooms in an office building to be located on an outside wall. In the design of electrical equipment some components may fail to function properly if located too far apart, while others fail when located too close together. An alternative to using the GPP to handle these complications would be to add constraints which simply require the distance between the locations containing objects be greater than (or less than) some fixed number.

a. Facilities Location

We begin by describing a simple example in which three plants a_i (i = 1, 2, 3) are to be assigned to three locations b_k (k = 1, 2, 3). The fixed costs f_{ik} of assigning plant a_i to location b_k are given in Table II.1. Table II.2 gives the number of units u_{ij} to be shipped from plant a_i to plant a_j, and Table II.3 gives the unit costs $d_{k\ell}$ of shipping from location b_k to b_ℓ.

Plant a_i \ Location b_k	1	2	3
1	200	300	250
2	400	350	300
3	600	700	650

Table II.1 Values of f_{ik}

Plant a_i \ Plant a_j	1	2	3
1	--	50	100
2	60	--	80
3	40	70	--

Table II.2, Values of u_{ij}

Location b_k \ Location b_1	1	2	3
1	--	2	3
2	1	--	2
3	3	3	--

Table II.3 Values of $d_{k\ell}$

We compute the C and H matrices described above:

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

$$
=\left[\begin{array}{ccc|ccc|ccc}
200 & - & - & - & 100 & 150 & - & 200 & 300 \\
- & 300 & - & 50 & - & 100 & 100 & - & 200 \\
- & - & 250 & 150 & 150 & - & 300 & 300 & - \\
\hline
- & 120 & 180 & 400 & - & - & - & 160 & 240 \\
60 & - & 120 & - & 350 & - & 80 & - & 160 \\
180 & 180 & - & - & - & 300 & 240 & 240 & - \\
\hline
- & 50 & 120 & - & 140 & 210 & 600 & - & - \\
40 & - & 80 & 70 & - & 140 & - & 700 & - \\
120 & 120 & - & 210 & 210 & - & - & - & 650
\end{array}\right]
$$

$$
H = \left[\begin{array}{ccccccccc}
1 & 1 & 1 & & & & & & \\
 & & & 1 & 1 & 1 & & & \\
 & & & & & & 1 & 1 & 1 \\
1 & & & 1 & & & 1 & & \\
 & 1 & & & 1 & & & 1 & \\
 & & 1 & & & 1 & & & 1
\end{array}\right]
$$

As before, for i = 1,2,3; k = 1,2,3; let

$$
x_{ik} = \begin{cases} 1 & \text{if plant } a_i \text{ is assigned to location } b_k \\ 0 & \text{otherwise} \end{cases}
$$

$x' = (x_{11}\ x_{12}\ x_{13},\ x_{21}\ x_{22}\ x_{23},\ x_{31}\ x_{32}\ x_{33})$, and

$e_6' = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)$.

The location problem is then

minimize $x'Cx$

subject to $Hx = e_6$

and x is a binary vector

It should be emphasized that in this problem the quantities shipped between plants, u_{ij}, were given and fixed, which in more general instances is not always the case. There are a number of complications which may be introduced. For example, the shipment of goods between plants, if placed in certain locations, may require the construction of a road or pipeline, and the fixed cost of such construction must be taken into consideration. It is also possible that more than one plant may be placed in one location. A very common problem is that we already have a number of plants built and wish to add a few more. Suppose we are considering n plants a_i and n locations b_k, but $s < n$ of these plants are already built. In particular, assume that for $j = 1,2,\ldots,s$ we have plant a_{i_j} built in location b_{i_j}. We can formulate this problem as a GPP by letting

$$R = \{(\{a_{i_1}\},\{b_{i_1}\}),\ (\{a_{i_2}\},\{b_{i_2}\}),\ \ldots\ ,\ (\{a_{i_s}\},\{b_{i_s}\})\}$$

A second more complicated example of the GPP is given, and the relationship of the 0-1 variables shown. Let $A = \{a_1,a_2,\ldots a_{10}\}$ be a set of ten plants and $B = \{b_1,b_2,\ldots,b_{10}\}$ be a set of ten locations. We are given that u_{ij} units are to be shipped from plant a_i to plant a_j, the cost of shipping one unit from location b_k to b_ℓ is $d_{k\ell}$, and the cost of placing plant a_i in location b_k is f_{ik}.

Let $A_1 = \{a_8, a_9, a_{10}\}$

$B_1 = \{b_8, b_9, b_{10}\}$

$A_2 = \{a_4, a_5, a_6, a_7, a_8\}$

$B_2 = \{b_1, b_2, b_3, b_4, b_5, b_8, b_9\}$

$A_3 = \{a_1, a_2, a_3, a_4, a_5, a_9\}$, and

$B_3 = \{b_2, b_7, b_8\}$

Let $R = \{(A_1,B_1), (A_2,B_2), (A_3,B_3)\} \subset P(A) \times P(B)$

We define C and H as before and obtain the problem:

$$\text{minimize} \quad x'Cx \qquad \text{II.1}$$

$$\text{subject to} \quad Hx = e_{20} \qquad \text{II.2}$$

$$x \text{ binary} \qquad \text{II.3}$$

and elements from A_1 must be placed in locations of B_1, and only those locations — II.4a

elements from A_2 must be placed in locations of B_2 — II.4b

elements placed in B_3 locations must come from A_3 — II.4c

Statements II.1, II.2, II.3 make up a QAP, and II.4abc tell us that

$$x_{ij} = 0 \text{ for } i \text{ and } j \text{ such that } \begin{cases} a_1 \in A_1 \text{ and } b_j \in B \setminus B_1 & \text{II.4a} \\ \text{or } a_1 \in A \setminus A_1 \text{ and } b_j \in B_1 & \\ a_1 \in A_2 \text{ and } b_j \in B \setminus B_2 & \text{II.4b} \\ a_1 \in A \setminus A_3 \text{ and } b_j \in B_3 & \text{II.4c} \end{cases}$$

(Here $A \setminus A_1$ is the complement of A_1 in A, etc.)

Interpretations of these constraints might be: plants of set A_2 require a water supply which is available only in locations of set B_2; locations of B_3 are small and must be occupied by plants not requiring much space, those of A_3; and plants of A_1 must be built together in a certain region B_1.

b. Building Layout

The building layout problem is quite similar to the facilities location problem except that we are concerned with what goes on between various rooms in a building rather than shipments between plants. One objective may be to minimize the time employees spend walking between offices or rooms by finding optimal locations for the offices. For

example, in the case of a hospital, we have a wide collection of employees such as surgeons, nurses, student nurses, etc., who collect various salaries and find it necessary (according to their job) to travel with varying degrees of frequency between different rooms such as an emergency room, operating room, scrub-up room, or sterilizing room. Now it has been pointed out that, on the average, people in a typical hospital spend 38 percent of their time walking between rooms, and taking relative salaries into account, this represents 34 percent of the cost for staff salaries. If we assume that careful planning can reduce this figure by one quarter, a savings equivalent to 8 1/2 percent of the cost for staff salaries is realized. This may amount to a substantial sum of money, especially over a period of many years (Whitehead and Elders [30]).

There are, however, some special problems in building layout which do not occur in any analogous manner in facilities location, control panel layout, or the wiring problem. We begin in a typical building layout problem by making a list of the rooms, and the frequency with which employees of different salaries travel between the rooms. For each pair of rooms a_i and a_j we may then compute a cost factor u_{ij} equal to the sum of the products of the number of people traveling between the two rooms and their salaries (say, in dollars/hours). The movements are then examined to see if they can be eliminated as unnecessary or new facilities or organization could lead to their elimination. For example, a recovery ward may be added so the surgeon or anaesthetist doesn't need to visit patients in their own room right after an operation. A set of location units b_k is determined. Since the rooms may vary in size, some may require only one location while others require several (adjacent locations). Instead of distance $d_{k\ell}$ between locations b_k and b_ℓ, we let $d_{k\ell}$ represent

the time (say, in fractions of hours) it takes to travel from b_k to b_ℓ. Of course, a difficulty arises when a room occupies several locations, and in this case we may arbitrarily take the shortest (or longest) distance between the locations used by the rooms.

Once a placement has been determined, refinements may be needed such as the addition of a corridor to help maintain aseptic conditions in the operating room, or even changes for aesthetic purposes. It should be pointed out then, that the solution of the building layout problem as a QAP (or GPP) yields only an approximate solution to the real problem. Some factors such as the cost of air ducts, plumbing, and wiring between rooms we have neglected, because in most instances the differences are small for various layouts.

c. Control Panel Layout

One of the objectives in designing an instrument or control panel may be to minimize eye travel distance. Assume we are assigning n instruments a_i to n locations b_k. Using notation similar to before, we let u_{ij} be the frequency of transitions from instruments a_i to a_j and $d_{k\ell}$ the Euclidean distance between locations b_k and b_ℓ. As before, the cost and assignment matrices C and H may be computed to obtain a QAP:

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_{2n}$$

$$x \text{ binary}$$

This problem is discussed in (Dorris [8, p. 18-21]), while a linear case is given in (Freund and Sadosky [9]).

d. The Wiring Problem

Beginning in the late 1950's, a problem associated with the construction of computers, now known as the backboard wiring problem, began to

receive attention (Loberman and Weinberger [20], and Steinberg [25]). The problem is to find a placement (assignment) for a number of electrical components, which must be interconnected, on a board so that the total length of wire used to make the connections is minimal. An assumption is made that minimal total wire length is the best approximation for satisfying a number of goals for good circuit design, which include:

reducing capacitance
reducing delay line effects
distribution of heat sinks
avoidance of excessive wire build-up on routing channels
avoiding cross-talk without excessive use of expensive shielding
neatness to reduce construction and maintenance costs
using smallest amount of wire

We suppose a set of n electrical components a_i are given which must be assigned to a set of n locations b_k. As before, without loss of generality, it is assumed the number of components equals the number of locations, and they will be assigned in a one-to-one manner. Let u_{ij} be the number of wires connecting components a_i and a_j. The distance $d_{k\ell}$ between b_k and b_ℓ will, of course, depend on the wiring channels used and may be either Euclidean or "street" distance. The cost and assignment matrices may be found, as before, to give the problem:

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_{2n}$$

$$x \text{ binary}$$

Example: The best known example, and, in fact, about the only real problem which appears to have been studied in detail by many authors, is the one given in (Steinberg [25], and (Graves and Whinston [15]). It is pointed out in (Hanan and Kurtzberg [10]) that there is a need for studying additional realistic problems. Here we give a small example just to illustrate the ideas involved, the structure of the matrices, and some

possible extensions. Consider the problem of placing four components a_i $(i = 1,2,3,4)$, with u_{ij} wire connections between a_i and a_j, in four locations b_k $(k = 1,2,3,4)$. The values of u_{ij} are given in Table II.4 and the distances $d_{k\ell}$ between locations b_k and b_ℓ are given in Table II.5. Figure II.1 shows the actual arrangement of locations on the board. We have used Euclidean distances to compute the $d_{k\ell}$.

i \ j	1	2	3	4
1	0	3	7	1
2	3	0	2	5
3	7	2	0	3
4	1	5	3	0

Table II.4 Values of u_{ij}

k \ l	1	2	3	4
1	0	1	1	$\sqrt{2}$
2	1	0	$\sqrt{2}$	1
3	1	$\sqrt{2}$	0	1
4	$\sqrt{2}$	1	1	0

Table II.5 Values of $d_{k\ell}$ with Euclidean Distance

b_1	b_2
b_3	b_4

Figure II.1

The assignment matrix H is

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & & & & & & & & & & & & \\ & & & & 1 & 1 & 1 & 1 & & & & & & & & \\ & & & & & & & & 1 & 1 & 1 & 1 & & & & \\ & & & & & & & & & & & & 1 & 1 & 1 & 1 \\ 1 & & & & 1 & & & & 1 & & & & 1 & & & \\ & 1 & & & & 1 & & & & 1 & & & & 1 & & \\ & & 1 & & & & 1 & & & & 1 & & & & 1 & \\ & & & 1 & & & & 1 & & & & 1 & & & & 1 \end{bmatrix}$$

The cost matrix C could be computed as before, but since the (u_{ij}) and $(d_{k\ell})$ matrices are symmetric, we may write C as the partitioned matrix

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

where $C_{ij} = \begin{pmatrix} ij \\ C \\ rs \end{pmatrix}$ $\quad r = 1,2,3,4$ $\quad s = 1,2,3,4$

and $C^{ij}_{rs} = \begin{cases} u_{ij} d_{rs} & \text{if } i < j \quad (i = 1,2,3,\ j = 2,3,4) \text{ and } r \neq s \\ 0 & \text{otherwise} \end{cases}$

So C is the 16 x 16 matrix whose non-zero elements are shown:

$$C = \left[\begin{array}{cccc|cccc|cccc} & & & & 0 & 3 & 3 & 3\sqrt{2} & 0 & 1 & 1 & \sqrt{2} \\ & & & & 3 & 0 & 3\sqrt{2} & 3 & 1 & 0 & \sqrt{2} & 1 \\ & & & & 3 & 3\sqrt{2} & 0 & 3 & 1 & \sqrt{2} & 0 & 1 \\ & & & & 3\sqrt{2} & 3 & 3 & 0 & \sqrt{2} & 1 & 1 & 0 \\ \hline & & & & 0 & 2 & 2 & 2\sqrt{2} & 0 & 5 & 5 & 5\sqrt{2} \\ & & & & 2 & 0 & 2\sqrt{2} & 2 & 5 & 0 & 5\sqrt{2} & 5 \\ & & & & 2 & 2\sqrt{2} & 0 & 2 & 5 & 5\sqrt{2} & 0 & 5 \\ & & & & 2\sqrt{2} & 2 & 2 & 0 & 5\sqrt{2} & 5 & 5 & 0 \\ \hline & & & & & & & & 0 & 3 & 3 & 3\sqrt{2} \\ & & & & & & & & 3 & 0 & 3\sqrt{2} & 3 \\ & & & & & & & & 3 & 3\sqrt{2} & 0 & 3 \\ & & & & & & & & 3\sqrt{2} & 3 & 3 & 0 \\ \hline & & & & & & & & & & & \end{array}\right]$$

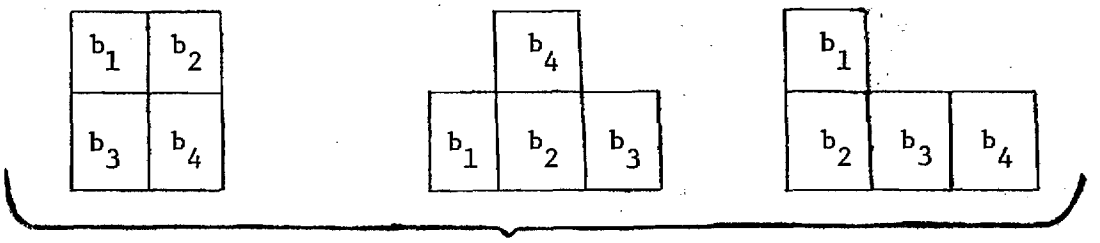
The problem is

$$\text{minimize} \quad x'Cx$$

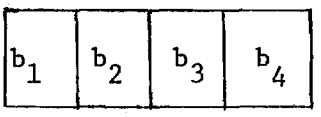
$$\text{subject to} \quad Hx = e_8$$

$$x \text{ binary}$$

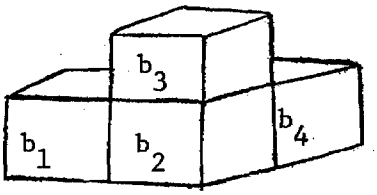
Some Extensions The above problem may be altered by introducing weights to the u_{ij}'s, as was done in the hospital example of the building layout problem, or by changing the $(d_{k\ell})$ matrix. We may wish to change the $(d_{k\ell})$ matrix to correspond to such physical arrangements of locations as



2 dimensional arrangements



1 dimensional



or possibly a 3 dimensional layout

If $\mathcal{D}$ is a set of matrices

$$\mathcal{D} = \{D = (d_{k\ell})\}$$

we may wish to consider the problem of finding the best arrangement of locations as well:

$$\begin{array}{lll} \text{minimize} & \text{minimize} & x'C_D x \\ D \in \mathcal{D} & Hx = e & \\ & x \text{ binary} & \end{array}$$

(Here C_D indicates the cost matrix depends on D).

As was done in the facilities location problem, we may use the GPP to require certain sets of components be located in specified sets of locations. This may be useful to keep the components of one section, say an amplification or power section, of a piece of electrical equipment together, as an aid to construction or maintenance.

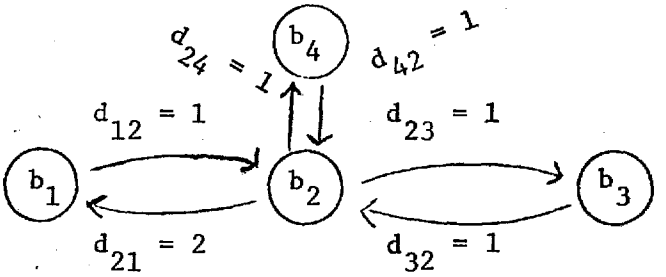
II.3 The Traveling Salesman Problem

Suppose a salesman is given a list of n cities connected by a number of roads of known distance. His problem, known as the traveling salesman problem, is to find a route of minimal length he can follow so as to visit each city once and only once, and return to where he started (Hu [17, pp. 270-271] and Garfinkel and Nemhauser [11, pp. 354-361]).

a. The Classical Traveling Salesman Problem

There are several different versions and formulations of the TSP. We are concerned with starting the problem as a QAP, but first mention, for clarification, some of the variations. First, we are considering what is called the closed tour version. This simply means that the salesman must end up where he started. An alternative, is the open tour version in which the salesman starts in a given city, visits each city once and only once, but does not return to the original city. It is easy to convert the closed tour version to the open tour, by dividing the starting city into two cities. We are just going to consider the closed tour here.

Now the following example is a simple TSP which has no solution. We are given four cities b_k (k = 1,2,3,4) and some roads of known distances $d_{k\ell}$ between them as shown below.



It is easy to see that it is impossible to visit all four cities and not visit b_2 three times. There is a modification of the TSP described in b. below which allows for the possibility of passing through cities, but as the classical TSP, this problem has no solutions. We also note that $d_{12} = 1$ and $d_{21} = 2$, so that $d_{12} \neq d_{21}$. It is not necessary that the distances between two cities be the same each way. The idea of a subtour comes from formulations of the TSP in which variables x_{ij} are used:

$$x_{ij} = \begin{cases} 1 & \text{if the salesman travels from } b_i \text{ to } b_j \\ 0 & \text{otherwise} \end{cases}$$

Unless constraints are added to prevent it, a "solution" may occur consisting of more than one trip, as shown below:

The trips 1-2-3-1 and 4-5-6-7-4 are called subtours, and obviously cannot be a solution to any problem in which a salesman must begin and end his trip in the same city. A tour is a sequence for visiting each city exactly once and ending up in the starting city. In the formulation we now give for the TSP as a QAP, the problem of subtours does not arise.

Suppose we are given n cities b_k $(k = 1,2,\ldots,n)$ with distances $d_{k\ell}$ from city b_k to b_ℓ. We assume $d_{k\ell} \geq 0$, and in cases where there is no $d_{k\ell}$ given (as when there is no direct road from b_k to b_ℓ) we assign some large number M. We think of the TSP as a facilities location problem in which n plants a_i $(i = 1,2,\ldots,n)$ are to be assigned to the n cities b_k $(k = 1,2,\ldots,n)$. The units shipped from a_i to a_j, u_{ij} are given by:

i) for $i = 1,2,\ldots,n-1$ $\quad u_{i\ i+1} = 1$

ii) for $i = n$ $\quad u_{n\ 1} = 1$

iii) all others $\quad u_{ij} = 0$

and we wish to minimize the total cost. Constructing cost and assignment matrices C and H as before, we have the TSP stated as the QAP

$$\begin{aligned} &\text{minimize} && x'Cx \\ &\text{subject to} && Hx = e_{2n} \\ & && x \text{ binary} \end{aligned}$$

A variable $x_{ik} = 1$ in the final solution x means city k is visited as the ith city.

b. Modifications of the Traveling Salesman Problem

As noted in a. above there are some TSP's which have no solution because it is impossible to visit each city exactly once. There are other examples in which we could reduce the distance traveled if we could "pass through" a city, i. e., visit it more than once. Such an example is shown below in Figure II.2.

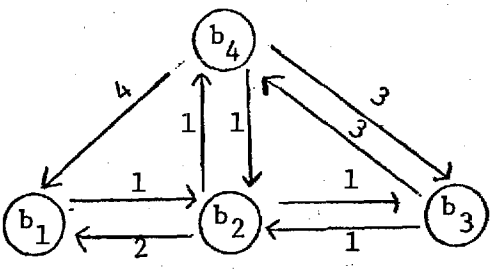


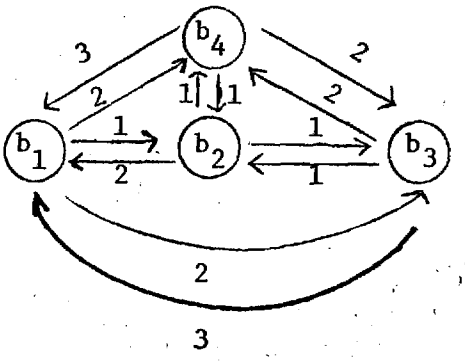
Figure II.2

The shortest route to visit all cities by the TSP is 1-2-3-4-1 and has length 9. If we are permitted to visit b_2 three times, we could choose the route 1-2-4-2-3-2-1 which has length 7. One method to avoid having

the larger solution to the TSP forced upon us, which would be desirable in most real applications, is to construct an expanded TSP as follows. Given n cities and some values $d_{k\ell}$ representing the distances of roads from city b_k to b_ℓ, define values of $\tilde{d}_{k\ell}$ for all $k = 1,2,\ldots,n$; $\ell = 1,2,\ldots,n$ by

$$\tilde{d}_{k\ell} = \begin{cases} \text{the shortest distance from city } b_k \text{ to } b_\ell & \text{if } k \neq \ell \\ \infty & \text{if } k = \ell \end{cases}$$

(We will need to keep track of the shortest paths when stating the solution of the original problem from the expanded problem). The example in Figure II.2 would then become

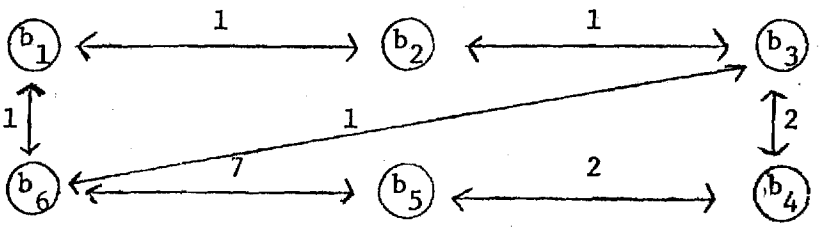


with the optimal solution 1-2-4-3-1 and distance 7. If the original problem in Figure II.2 had satisfied the triangle inequality for distances*

$$d_{ij} + d_{jk} \geq d_{ik}$$

and we had been given values for <u>all</u> $d_{k\ell}$ ($k = 1,2,\ldots,n$; $\ell = 1,2,\ldots,n$), then the optimal routing wouldn't require visiting b_2 three times and the expanded TSP wouldn't have been needed (Hu [17, pp. 270-3]). The following example is one in which the given $d_{k\ell}$ satisfies the triangle inequality, but a city is visited twice to obtain the shortest route 1-2-3-4-5-4-3-6-1 (length 12). The reason here is that not all the inter-distances among the cities are given.

*It doesn't: $2 = 1 + 1 = d_{3,2} + d_{2,4} < d_{3,4} = 3$



We now mention some extensions of the TSP. In addition to being given n cities b_k (k = 1,2,...,n) and some distances $d_{k\ell}$ between cities b_k and b_ℓ, we may be given times $t_{k\ell}$ between cities b_k and b_ℓ and due dates τ_k by which a city b_k must be visited. For a given tour ψ, let $b_{\psi(i)}$ be the ith city visited, and suppose $b_{\psi(i)}$ is visited at time

$$\hat{\tau}_{\psi(i)} = t_{\psi(0)\psi(1)} + t_{\psi(1)\psi(2)} + \cdots + t_{\psi(i-1)\psi(i)}$$

for i = 1,2,...,n. If we define the tardiness in visiting a city as $\hat{\tau}_k - \tau_k$, one objective may be to find a tour ψ which minimizes the maximum tardiness. Another problem may be to minimize a linear combination of the distance traveled and the maximum tardiness. The GPP may be used to make such requirements as the first city visited must be one of three selected cities, or the last two cities visited must include a certain given city.

In another modification of the TSP, considered by (Srivastava et. al., [24]), the set of cities $B = \{b_1, b_2, \ldots, b_n\}$ is divided into a home city and a family of s collectively exhaustive and mutually exclusive sets of cities: $B = \{\text{home city}\} \cup \bigcup_{i=1}^{s} B_i$. The problem is to determine an optimal tour, starting at the home city, which includes at least one city from each of the sets B_i (i = 1,2,...,s). An application of this modification, in which a salesman has to complete multiple jobs subject to competition, is given in (Garg, et. al., [12]).

c. Production Scheduling Application

Suppose we have n tasks b_k (k = 1,2,...,n) which are to be performed on some machine. A change over time $d_{k\ell}$ is required to change the machine from doing task b_k to task b_ℓ. The problem of selecting a sequence in which to perform the tasks and return the machine to its original state so the sum of changeover times is minimized is a TSP (Maxwell [21]).

An n-job, m-machine scheduling problem is modelled as a TSP in (Wismer [31]).

d. The Multi-Traveling Salesmen Problem, Its Generalizations, and Some Applications

The TSP is actually a special case of the multi-traveling salesmen problem (M-TSP) (Svestka and Huckfeldt [26]). In the M-TSP we are given n cities b_k (k = 1,2,...,n), one of which is designated as a "home city", and distances $d_{k\ell}$ between cities b_k and b_ℓ. There are m salesmen instead of one, and the problem is to determine routes for each of the m salesmen to follow so that every city (except the home city) is visited exactly once by exactly one salesman, the home city is included in each salesman's route, and the total distance traveled by all salesmen is a minimum. We note that in the M-TSP a "home city" is designated beforehand. Three alternatives are to designate several home cities, as is done in the multi-terminal delivery problem (Tillman and Hering [27]), or allow for either one or several undesignated home cities which will be determined as part of the problem. (Not requiring a home city actually falls into the last two cases). Another generalization of the M-TSP is given in (Zak [32]) where each salesman may have a different distance matrix.

We now give some applications of the M-TSP and its generalizations.

The Bank Messenger Scheduling Problem

In this problem a crew of messengers collects deposits at branch banks and returns them to a central office for processing. If there are m messengers and n branches, this is the M-TSP. The Cleveland Trust Company, Cleveland, Ohio, has actually made use of the M-TSP to determine routes for its messengers (Svestka and Huckfeldt [26]).

Printing Press Scheduling for Multi-Edition Periodicals

A printing press scheduling problem is given as a 3-traveling salesmen problem with a number of side constraints in (Gorenstein [14]). But even a simplified version of this problem is not solvable for problem sizes with practical applications.

Scheduling Excavators Movements

The scheduling of the movements of excavators in mine pits is given in (Tsoi, et. al., [28]) as a M-TSP. Cases with route length and cycle constraints are given.

Single Terminal Delivery Problem

In this problem delivery trucks supply customers from a single terminal. This is a M-TSP, with the restriction that the trucks have limited capacity and need to return to the home city (terminal) to refill.

Multiterminal Delivery Problem

This is an application of the generalization of the M-TSP in which several home cities are designated. While in the single terminal delivery problem we assign customers to routes so that the total distance is minimized, in the multiterminal delivery problem we need to assign customers to routes and routes to terminals. This problem and references for the single terminal case may be found in (Tillman and Hering [27]).

II.4 Applications of the QSCP and Related Problems

In this section we give a number of examples of set covering problems, most of which were originally formulated as LSCP (or LSPP). These include the delivery problem, location of fire hydrants, airline crew scheduling, assembly line balancing problem, information retrieval, project selection, capital budgeting, and attack/defense networks. A general structure common to all of these applications is first given, then each application is considered separately. The project selection problem is considered in detail and some directions are indicated for the formulation of a more realistic model.

We recall that the QSCP was

$$\begin{aligned} &\text{minimize} && x'Cx \\ &\text{such that} && Ax \geq e_m \\ & && x \text{ is a binary vector} \end{aligned}$$

where x, C and A were described in II.1. A was an m x n matrix. In the applications each of the m rows is identified with either a client (delivery problem), a flight segment (airline crew scheduling), a task (assembly line balancing), or some activity. Various subsets of the clients flight segments, tasks, or activities are selected to form (respectively) routes, trips, assembly stations, or projects, which are identified with the columns of A. For example, in the delivery problem, if client i is included in route j, then $a_{ij} = 1$, if not $a_{ij} = 0$. e_m is a vector of ones indicating that all clients must be included in the final solution. In the set covering problem, a client may be served more than once. If we want to restrict a client to one service in a problem, the partitioning problem would be used. In the LSCP and LSPP, costs are assigned to the routes, trips, assembly stations, or projects (i. e., the columns of A).

The quadratic versions of these problems permit additional costs or discounts for various pairs of columns selected. In some applications there are good reasons why two activities should cost less than the sum of the costs of the activities individually, while in other cases there appears to be no reason for extending the linear formulation to a quadratic one.

a. Routing Problems: Truck Delivery and School Bussing

In the truck delivery and school bussing problems we are concerned with the delivery or pick-up of items, or, in the case of school bussing, children. There are a number of delivery or pick-up stations, each of which must be visited at least once, and a central terminal or school from which the items are sent out or collected. Various subsets of the delivery stations are formed to make up routes, and each route has a cost associated with it. The problem of selecting a set of routes so that each station is included in at least one route is a LSCP, or, if reasons can be found for quadratic interaction of the costs, a QSCP. In the case of school bussing one reason to use a QSCP might be to help achieve a racial balance. The question remains of how to do this. This would allow certain combinations of routes to be desirable and others to be undesirable. Since trucks or buses have limited capacity, the number of items picked up or delivered on a route is limited; also, since the total number of available trucks is limited (not allowing multiple trips for a vehicle), we may need a solution in which the total number of routes selected is less than or equal to the total number of vehicles (Balinski and Quandt [33] and Murty [22]).

b. Location of Fire Hydrants

The problem of locating fire hydrants is discussed in (Murty [22]). We consider a network whose arcs are streets, and nodes are intersections

of the streets. The problem is to select a subset of the nodes for placement of fire hydrants so each street has at least one fire hydrant. This problem is probably best left as a LSCP.

c. Airline Crew Scheduling

In the airline crew scheduling problem flight segments are selected to form trips, and the idea is to select trips to cover all flight segments. This is a LSCP. See (Murty [22] and Arabeyre, et. al., [2]).

d. Assembly Line Balancing

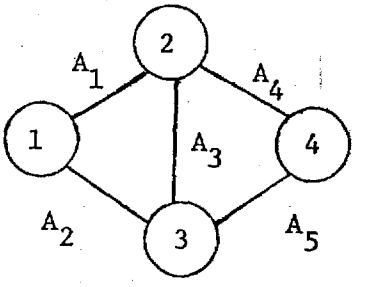
Tasks are combined to be performed at assembly stations in the assembly line balancing problem. The problem is to determine which assembly stations are to be set up. This is a partitioning problem, since a task is to be performed exactly once. Complications arise in the form of precedence relations on tasks and due dates by which tasks must be performed. (Murty, [22] and Salvesan [23]).

e. Information Retrieval

The information retrieval problem is how to select a number of tapes from a collection of tapes, each of which contains subsets of required information, so that all desired information may be found by searching a minimal length of tape. This is a LSCP and there appears to be no advantages in considering it as a QSCP. See (Day [7]).

f. Attack/Defense Networks

Attack and defense of networks may be formulated as QSCP. Consider the network shown below



The nodes in the network are thought of as strategic points; while the arcs represent communication lines such as roads, bridges, pipelines, etc. The objective is to attack the network in a optimal manner so that nodes 1 and 4 are separated. If A_1 represents arc i and x_i the associated variable, then $x_i = 1$ means A_i is destroyed and $x_i = 0$ means A_i is not destroyed. The constraint matrix which requires every path from node 1 to node 4 be cut at least once is

$$\begin{array}{c} \begin{matrix} A_1 & A_2 & A_3 & A_4 & A_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

If a cost is assigned to destroying each arc, a linear objective function may be obtained to give a LSCP. However, the cost of destroying two arcs, say A_1 and A_5, may be greater than the sum of the costs of destroying them individually. For example, they may be far apart and require some scarce resource to destroy. In this case a quadratic objective function $x'Cx$ is used to consider such interactions. The element c_{15} would be positive and represent the additonal cost of cutting A_1 and A_5. The attack problem is a good example of a LSCP which is better modelled as a QSCP. The defense problem, how to optimally defend a network which will be attacked, by the construction of more arcs subject cut limitations, is also a QSCP. (Bellmore and Ratliff [4,5] and Jarvis [18]).

g. Project Selection and Capital Budgeting

We consider the problem where a number of projects are combined to form activities. A set of activities is to be selected in an optimal

manner so that all required projects are included among the selected activities. Since it would probably be desired that a project is not included more than once, a partitioning problem would be used. Since there may be advantages of selecting certain pairs of activities, a QSCP is used.

To give some insight into the discount cost arising from the selection of two activities, consider the interaction which occurs in construction activities. There are three primary reasons why two activities are sometimes cheaper than the sum of the individual costs:

1. a materials discount When two activities contain projects which use the same type of material, a savings may be realized by buying this material in bulk. There is a limit to this savings, e. g., three $5,000 projects may lead to a savings, but the addition of two more $5,000 projects may not be significant.
2. already having equipment on the construction site To move a piece of machinery may cost $400, $100 to get a truck on the site and $300 to move the machine. Perhaps three machines could be moved at the same time for 100 + 3(300) = $1000 or a $200 savings over 3(400) = $1200.
3. insurance savings (four jobs cost only a little more than one).

Unfortunately, there are also three major reasons why the QSPP doesn't apply in many cases. First, we assume we know all of the projects (and activities) from the start, while in real problems new projects arise as work proceeds on old ones. Then we assume all the projects are funded ahead of time and from one source. In practice, funding comes from several different sources, cannot be mixed, and is in

various stages of uncertainty. Finally, as in construction work, there is a scheduling problem. We may need to hire two or three contractors at once, because one may not be able to simultaneously work on three projects which must be done by a certain date.

In practice then, a person with, say three, projects would probably

1. do none of them
2. do them all together, if possible
3. do separately as required by funding and scheduling.

Additional complications may occur in the project selection/capital budgeting problem in that

1. the selection of one activity may either necessitate or preclude the possibility of the selection of another activity
2. the total number of activities is limited
3. the total amount of money is limited (perhaps cancelling some optional activities).

It appears, in view of the difficulties described above, that a new model which takes the scheduling and funding, at least, into account may be developed. (Alexander [1] and Weingartner [29]).

h. Political Districting

In political districting, an area (e. g., a state) is partitioned into a number of smaller areas called districts which are assigned a number of representatives. The area has also been partitioned into a number of basic population units (e. g., counties). A political districting plan consists of assigning the population units to the districts so that:

i) a population unit is assigned to exactly one district

ii) the total population assigned to a district doesn't deviate from the mean district population by more than a given number

iii) a district is contiguous, i. e., it is possible to walk between population units making up the district without going into another district (roughly speaking)

and iv) a district is compact, i. e., somewhat circular or square

Details and examples are given in (Garfinkel and Nemhauser [10]). Without the contiguity and compactness restrictions, this problem may be formulated as a QSCP. Suppose we have n basic population units a_i (i = 1,2,...,n) with the population of unit a_i being d_i, and m districts b_k (k = 1,2,...,m). Define variables x_{ik} by

$$x_{ik} = \begin{cases} 1 & \text{if } a_i \text{ is assigned to } b_k \\ 0 & \text{otherwise} \end{cases}$$

and let $x' = (x_{11}\ x_{12} \cdots x_{1m};\ x_{21}\ x_{22} \cdots x_{2m};\ \cdots\ ;\ x_{n1}\ x_{n2} \cdots x_{nm})$

The QSCP is then:

$$\text{minimize} \quad x'Cx$$

$$\text{subject to} \quad Hx = e_n$$

$$\text{and } x \text{ a binary vector}$$

where C is the $(n \cdot m) \times (n \cdot m)$ partitioned matrix

$$C = \left[\begin{array}{cccc} C_{11} & C_{12} & \cdots & C_{1m} \\ \hline C_{21} & C_{22} & \cdots & C_{2m} \\ \hline \vdots & & & \\ \hline C_{m1} & & & C_{mm} \end{array}\right]$$

$$C_{ij} = \begin{bmatrix} d_1^2 - 2dd_1 & d_1d_2 & \cdots & d_1d_n \\ d_2d_1 & d_2^2 - 2dd_2 & & d_2d_n \\ \vdots & \vdots & & \\ d_nd_1 & d_nd_2 & \cdots & d_n^2 - 2dd_n \end{bmatrix} \quad \text{if } i = j$$

and C_{ij} = the n x n zero matrix for $i \neq j$

(here $d = \frac{1}{m} \sum_{i=1}^{n} d_i$, the mean district population)

and H is the n x (n · m) matrix

$$H = \begin{bmatrix} e_m' & & & \\ & e_m' & & \\ & & \ddots & \\ & & & e_m' \end{bmatrix}$$

II.5 Miscellaneous Applications

The coloring problem (Bessiere [6]), determining the minimum number of colors needed to color a map so that no two adjacent areas have

the same color, is a LSPP. (Lawler [19]) discusses the problem of minimizing "latency" in magnetic drum on disc storage computers as a QAP. The reader may see applications to staff and class scheduling, constrained least squares estimation, or message switching centers.

II.6 Some Comments on the Practicability of QSCP Applications

In this section we discuss the realistic values of the applications given in sections II.2, II.3, and II.4. Quite often, simple examples are used to illustrate the application of some theory, and, while they serve the purpose of clarifying the theory, they are of no practical use. Basically there are three reasons for this:

i) the real problem is too simple to waste time applying a complicated theory

ii) the theory is not complicated enough to model the real problem

iii) the theory is OK for the real problem but leads to computationally infeasible problems.

We expect the first case (i) to hold for many facilities location and building layout problems. For example, a motel is a relatively uncomplicated structure and probably wouldn't require a QAP to determine a layout. The true value of the QAP is apparent when a complicated structure such as a hospital is designed or a complex wiring problem is considered. Unfortunately, in these cases the problem may become computationally infeasible (iii).

A good example where (ii) the theory was not complicated enough is the project selection problem of II.4.g. Many of the references cited state that computationally infeasible problems result from the application of the theory described to real problems: (Jarvis [18, p. 88]) states

that although the attacker problem can be handled in a reasonably efficient manner, the defender problem cannot. Similar statements occur in (Maxwell [21]) concerning production scheduling, in (Gorenstein [14]) on printing press scheduling, and in (Freund and Sadosky [9]) on control panel layout. In other words, there are a large number of valuable applications, once efficient solution procedures are developed. These procedures may include new algorithms and combinations/modifications of old ones, or the attempt to include more complications of the original problem as in the GPP.

How about the use of quadratic versus linear models for some applications? As was mentioned in section II.4 some problems, e. g., the information retrieval problem, are naturally linear and no basis was found to consider quadratic interaction. Examples of applications of set covering problems with quadratic objective functions, with the exception of the special case of the QAP, are not easy to find. Most examples considered, which were not linear, such as construction project selection, exhibit higher order interactions. It does not appear that examples with high order interactions can be approximated by quadratic interactions in a successful manner. Although (Weingartner [29]) points out the tremendous advantage of the quadratic over the linear model for project selection/capital budgeting, some concrete, real applications remain to be found which can be solved as, say, a QSPP.

REFERENCES

1. Alexander, D., Interview, Physical Plant, Georgia Institute of Technology, 1973.

2. Arabeyre, J. P., J. Fearnley, F. C. Steiger, and W. Teather, "The Airline Crew Scheduling Problem: A Survey," Transportation Science 3, 140-163, 1969.

3. Balinski, M. L. and R. E. Quandt, "On an Integer Program for a Delivery Problem," Operations Research 13, 300-304, 1964.

4. Bellmore, M. and H. D. Ratliff, "Set Covering and Involutory Bases," Man. Sci. 18, 194-205, 1971.

5. Bellmore, M. and H. D. Ratliff, "Optimal Defense of Multi-Commodity Networks," Man. Sci. 18, 174-185, 1971.

6. Bessiere, F., "Sur la Recherche der Nombre Chromatique d'Un Graph par une Programme Lineaire en Nombres Entiers," Rev. Franc, Recherche Operat. 9, 143-148, 1965.

7. Day, K. H., "On Optimal Extracting from a Multiple File Data Storage System: An Application of Integer Programming," Operations Research 13, 482-495, 1965.

8. Dorris, A. L., "The Utility of Optimization Techniques in the Design of Man-Machine Systems," Master's Thesis, Georgia Institute of Technology, 1971.

9. Freund, L. E. and T. L. Sadosky, "Linear Programming Applied to Optimization of Instrument Panel and Workplace Layout," Human Factors 9, 295-300, 1967.

10. Garfinkel, K. S. and G. L. Nemhauser, "Optimal Political Districting by Implicit Enumeration Techniques," Man. Sci. 16, 495-504, 1970.

11. Garfinkel, K. S. and G. L. Nemhauser, Integer Programming, John Wiley & Son, 1972.

12. Garg, K. C., S. Kumar, P. Dass, and P. Len, "Generalized Traveling Salesman Problem Through n Sets of Nodes in a Competitive Market," Ablanf and Plassungsforschung (Germany) 11, 116-120, 1970.

13. Geddes, P. H., "The Solution of Large Size Quadratic Set Covering Problems," Master's Thesis (in progress), School of Industrial and Systems Engineering, Georgia Institute of Technology.

14. Gorenstein, S., "Printing Press Scheduling for Multi-Edition Periodicals," Man. Sci. 16, 373-383, 1970.

15. Graves, G. W., and A. B. Whinston, "An Algorithm for the Quadratic Assignment Problem," in Integer and Nonlinear Programming, ed. by J. Abadie, North Holland Pub. Co., Amersterdam, 473-497, 1970.

16. Hanan, M., and J. M. Kurtzberg (1972), "A Review of the Placement and Quadratic Assignment Problems," SIAM Review 14, 324-342, 1972.

17. Hu, T. C., Integer Programming and Network Flows, Addison-Wesley, 1969.

18. Jarvis, J. J., Optimal Attack and Defense of a Command and Control Communications Network, Ph.D. Dissertation, The Johns Hopkins University, 1968.

19. Lawler, E. D., "Notes on the Quadratic Assignment Problem," Harvard Computation Laboratory, unpublished paper, 1960.

20. Loberman, H., and A. Weinberger, "Formal Procedures for Connecting Terminals with a Minimum Total Wire Length," Journal of the Association of Computing Machinery 4, 428-437, 1957.

21. Maxwell, W. L., "The Scheduling of Single Machine Systems: A Review," The International Journal of Production Research 3, 177-199, 1964.

22. Murty, K. G., "On the Set Representation and Set Covering Problems," in Symposium on the Theory of Scheduling and its Applications, ed. by M. Beckman, G. Goos, and H. P. Kunzi, Springer-Verlag, 143-163, 1973.

23. Salvesan, M. E., "The Assembly Line Balancing Problem," Journal of Industrial Engineering 6, 18-25, 1955.

24. Srivastava, S. S., S . Kumar, K. C. Garg, and P. Len, "Generalized Traveling Salesman Problem through n Sets of Nodes," CORS Journal 7, 97-101, 1969.

25. Steinberg, L., "The Backboard Wiring Problem: A Placement Algorithm," SIAM Review 3, 37-50, 1961.

26. Svestka, J. A. and V. E. Huckfeld, "Computational Experience with an M-Salesman Traveling Salesman Algorithm," Man. Sci. 19, 790-799, 1973.

27. Tillman, F. A., and R. W. Hering, "A Study of a Look-Ahead Procedure for Solving the Multiterminal Delivery Problem," Transportation Research 5, 225-229, 1971.

28. Tsoi, A. K. H., S. M. Taskai, and V. N. Druzhinin, "The Scheduling of Movements of Excavators as a Problem of Several Traveling Salesmen," Trudy Instituta Gormogo Dela, Akademirijo Nauk Kazakhrkogo SSR (U. S. S. R.) 45, 9-13, 1971.

29. Weingartner, H. M., "Capital Budgeting of Interrelated Projects: Survey and Synthesis," Man. Sci. 12, 458-516, 1966.

30. Whitehead, N. and M. Z. Elders, "An Approach to the Optimum Layout of Single-Story Buildings," Architects' Journal 139, 1373-1380, 1964.

31. Wismer, D. A., "Solution of the Flowshop-Scheduling Problem with no Intermediate Queues," Operations Research 20, 689-697, 1972.

32. Zak, Yu. A., "Solution Algorithms for 'n Traveling Salesmen' Problem," Kibernetika (U. S. S. R.) 1, 99-106, (Russian, English Summary), 1972.

III. THE QUADRATIC SET COVERING PROBLEM

In this section we will develop a branch and bound scheme for solving quadratic set covering problems which arise in the context of layout design.

III.1 Introduction

Consider the following quadratic set covering problem.

$$\text{Minimize} \quad x'Cx$$

$$\text{subject to} \quad Ax \geq e$$

$$x \text{ is a binary vector}$$

Several procedures for solving problems with the above structure exist in the literature. In [2] Bazaraa and Goode presented a cutting plane algoritm for solving such problems. The algorithm excludes integer and noninteger points at each step, and stops with an optimal solution after a finite number of steps. The algorithm is suitable for problems with:

-- the number of variables $n \leq 40$

-- the number of constraints $m \leq 60$

-- low density of A and C

In [3], an implicit enumeration algorithm which extends the notions in [1] and [6] to the quadratic case, is developed for the above problem. Problems with up to 100 variables and 100 constraints, and small density of the cost matrix C can be handled by the procedure. Here, rather than attempting to develop an algorithm for solving any quadratic set covering problem with the above structure, we will develop a procedure for the specific applications of layout design. Existing layout procedures are heuristic in nature and can be classified into the following two categories:

a. Construction algorithms The layout is constructed from scratch by successively selecting a new object (department, room, facility,

etc.) until the layout is completed. These algorithms are usually based on qualititative evaluation of the necessity or nonnecessity of adjacency of different objects. CORELAP [8], ALDEP [9], and LSP [11] are examples of such constructive algorithms.

b. Improvement algorithms Given an existing layout, an improved layout is sought by interchanging the positions of various objects. The improvement is based on reducing the total flow or cost of flow. CRAFT [4] and the method of Whitehead and Elder [10] are examples of such algorithms.

For more details the reader may refer to Francis and White [5, Chapter 3]. Here we present a quadratic set covering formulation of the layout problem. The formulation handles single or multi-story buildings, objects with regular or irregular shapes, designing a layout from scratch, or adding new objects to an existing layout. The final layout is provided by a branch and bound optimization procedure.

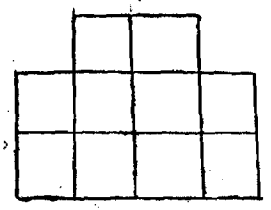
In order to produce an optimal layout the following input data are required. First, the size and required shape (or shapes) of each object must be provided. Secondly, candidate locations of each object are determined. If applicable, a fixed cost of assigning an object to a certain location is given. Finally, the flow (or interaction) between each two objects is determined. If not applicable or hard to estimate, this information may be substituted by the desirability or nondesirability of closeness of any two objects.

III.2 Formulation of the Problem

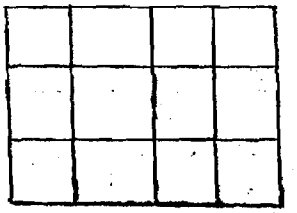
Suppose that we have an area, such as a warehouse, a floor (or multi-floors) of a building, a hospital, etc. Different objects must be

located in the area, such as rooms, manufacturing departments, facilities, etc. An interaction u_{ij} between objects i and j is realized. This interaction may be information flow, material flow, employee movement, etc. The problem is to locate the objects in such a way to minimize the total interaction weighted by the distance, plus any fixed costs that may be realized.

The problem is similar to the quadratic assignment problem, but has some significant differences. In contrast to the quadratic assignment formulation, where each object can occupy a single location, the objects here typically have different areas and designs, as illustrated below.



Object A



Object B

In order to accomodate this situation, the total area is divided into small units or blocks. The possible shapes and areas (in terms of numbers of blocks) of various objects are determined by the layout analyst. The analyst also determines a set of candidate locations for each object.

To illustrate, suppose that the total area is divided in 60 blocks as shown below. The analyst may decide that object A may occupy any one of the following candidate locations:

1. 1, 2, 3, 4, 11, 12, 13, 14, 22, 23
2. 10, 20, 30, 40, 9, 19, 29, 39, 18, 28
3. 55, 56, 57, 58, 45, 46, 47, 48, 36, 37
4. 38, 39, 27, 28, 29, 30, 17, 18, 19, 20
5. 21, 31, 41, 51, 22, 32, 42, 52, 33, 43
6. 14, 15, 23, 24, 25, 26, 33, 34, 35, 36

Each of the above set of blocks constitutes a candidate location of object A. Allowing the analyst to introduce candidate locations for each object helps eliminate undesirable locations* from the very beginning, takes advantage of the analyst experience, and meanwhile reduces the computational effort since the search space will be reduced.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60

Let x_{ij} be 1 if object i is assigned to its jth location, and 0 otherwise. Let $I(i)$ be the total number of possible locations of object i, and $J_i(k)$ be the set of blocks occupied by object i if it were assigned to its kth location. u_{ij} denotes the interaction between objects i and j and f_{ik} is the fixed cost (if any) of assigning object i to its kth position. The problem is formulated below, where $d(k_i,l_j)$ is the distance between the centroids of the kth and lth locations of objects i and j.

*For example an entrance to the building cannot be located in the interior of the building.

$$\text{Minimize} \quad \sum_{l=1}^{I(j)} \sum_{k=1}^{I(i)} \sum_{j=1}^{m} \sum_{i=1}^{m} u_{ij}\, x_{ik}\, x_{jl}\, d(k_i, l_j) + \sum_{k=1}^{I(i)} \sum_{i=1}^{m} f_{ik}\, x_{ik}$$

$$\text{Subject to} \quad \sum_{k=1}^{I(i)} x_{ik} = 1 \qquad \text{for } i = 1, 2, \ldots, m$$

$$\sum_{k=1}^{I(i)} \sum_{i=1}^{m} \alpha_{ikt}\, x_{ik} \leq 1 \qquad \text{for each block } t$$

$$x_{ik} \quad 0 \text{ or } 1 \qquad \text{for each } k = 1, 2, \ldots, I(i)$$

$$\text{for each } i = 1, 2, \ldots, m$$

where α_{ikt} is 1 if block $t \in J_i(k)$ and 0 otherwise.

Some Comments

a. The above formulation results in a quadratic set covering problem, where the constraints are of the set covering (partitioning) type and the objective function is quadratic. The first set of constraints requires that each object be assigned to one location and the second set insures that each block is occupied by at most one object.

b. Since the kth location of object i generally consists of many blocks, the distance between locations is taken to be from centroids of the locations. Another method to handle this difficulty is to divide u_{ij} among the blocks occupied by objects i and j., i. e., $\bar{u}_{ij} = u_{ij}/I(i) \times I(j)$. Then the weighted interaction is calculated by shipping $\bar{u}_{ij}$ units of flow from each block of object i to each block of object j, weighted by the distance between the corresponding blocks. For more detailed discussion refer to [10].

c. The analyst must be careful while estimating u_{ij}'s, since the flow among different objects may have different units. For example, different types of material are encountered in a manufacturing department, employees with a wide range of salary scale are encountered in a hospital, and so forth. Therefore, a common scale must be devised to obtain uniformity in the estimation of u_{ij}'s. For more details refer to [10].

In some cases either no flow between objects i and j is realized, or else it is difficult to make an accurate estimation of u_{ij}. In this case, the analyst may classify the desirability of closeness of any two objects as follows. Given objects i and j, the analyst determines if their adjacency or closeness is highly desirable, desirable, immaterial, undesirable, or highly undesirable. A subjective quantification of u_{ij} for these cases can be made, e. g., 200, 100, 0, -100, and -200 respectively.

d. Some design restrictions which prohibit certain combinations of assignments can be introduced. To illustrate, the constraint $x_{12} + x_{27} \leq 1$ would prevent the mutual assignment of the first and second objects to their second and seventh locations, respectively. Such constraints are important in some instances, e. g., objects 1 and 2 may be mutually flammable, and their respective 2nd and 7th locations are adjacent.

III.3 A Branch and Bound Scheme

We will now describe a branch and bound scheme for finding a layout with minimal objective. At each step of the algorithm a partial layout is at hand where a set of objects are assigned to some locations. A lower bound B on the cost of all possible completions of this partial layout (solution) is calculated. If B is less than the cost C* of the

best available layout known so far, we proceed by trying to assign a new object. Otherwise the partial solution is fathomed, the last assignment is prohibited, and a new assignment is sought. The method of calculating the lower bound and the progress of the search of the decision tree completely define the procedure.

Calculation of the Lower Bound

Suppose that some objects indexed by the set I have already been assigned. In particular, object i is assigned to its ν_i possible location. A lower bound B on the cost of all possible completions of this partial layout is given by $C_1 + C_2 + C_3$, where:

C_1 = Interaction cost among already assigned objects

C_2 = Lower bound on the interaction cost from unassigned objects to assigned objects

C_3 = Interaction cost among unassigned objects

Calculating C_1

$$C_1 = \sum_{\substack{j \in I \\ j \neq i}} \sum_{i \in I} u_{ij} x_{i\nu_i} x_{j\nu_j} d(\nu_{i,i}, \nu_{j,j}) + \sum_{i \in I} f_{i\nu_i}$$

Calculating $C_2 + C_3$

First we find a lower bound of the cost of placing the ith unassigned object in its νth location. This is the sum of $f_{i\nu}$, the cost between object i and the assigned objects, and a lower bound on the cost from other unassigned objects to object i. The first two costs can be determined exactly.

A lower bound on the cost from other unassigned objects to object i in its νth position can be calculated by iteratively finding the position of each unassigned object which minimizes its weighted interaction with object i, and then adding these minimal weighted interactions. Let the

sum of the above three costs be $b_{i\nu}$. The $C_2 + C_3$ is found by solving the following problem.

$$C_2 + C_3 = \text{Minimum} \quad \sum_{i \notin I} \sum_{\nu \in A_i} b_{i\nu} \, x_{i\nu}$$

$$\text{Subject to} \quad \sum_{\nu \in A_i} x_{i\nu} = 1 \qquad i \notin I$$

$$\sum_{i \notin I} \sum_{\nu \in A_i} \alpha_{i\nu t} \, x_{i\nu} \leq 1 \qquad \text{for each unassigned block } t$$

$$x_{i\nu} \text{ is either 0 or 1}$$

where:

A_i is the collection of available locations of object i which have not been eliminated by assigning the objects in the set I, and which have not been eliminated in the branch and bound search

$\alpha_{i\nu t}$ is 1 if block $t \in J_i(\nu)$ and 0 otherwise

The above problem is a linear set covering problem. The first collection of constraints requires that each object be assigned to one of its available locations, and the second set of constraints requires that each vacant block be occupied by at most one unassigned object. The lower bound $B = C_1 + C_2 + C_3$ is now available. Note that after the above problem is solved, a feasible solution to the overall problem is found and its quadratic cost is evaluated. The best layout C* is updated if necessary. Needless to say if the above linear set covering problem admits no feasible solutions, then the current partial solution is fathomed.

There are various ways of simplifying the above problem. For example, we can ignore the second set of constraints. The problem decomposes into a set of trivial knapsack problems, which amounts to independently finding the location of each unassigned object which

minimizes its cost with other objects. This method is adopted here. Another simplification of the above problem is to ignore the integrality restriction of the variables $x_{i\nu}$'s and solving a continuous linear programming problem instead of the linear set covering problem. Of course, these simplifications would generally result in smaller lower bounds thus reducing the speed of fathoming, but on the other hand would result in smaller computational effort per iteration.

The Branch and Bound Scheme

We will now describe a branch and bound scheme for finding an optimal layout. We assume familiarity of the reader with general branch and bound schemes. If this were not the case, the reader may refer to (Lawler and Wood [7]).

1. Order the objects, according to the sum of the interactions, say Let $C^* = \infty$. Choose the first object i_1 and place it in a suitable location ν. Let A_i be the available locations of object i. Let $I = \{i_1\}$, $k = 1$, and go to Step 2.
2. Calculate a lower bound B, where $B = \infty$ if no feasible completions exist. If $B \geq C^*$ go to Step 3. Otherwise go to Step 4.
3. The last assignment of object i_k to its ν_kth location is prohibited. Try to assign i_k to another vacant location. If this is possible replace ν_k by this new location and go to Step 2, otherwise free all the locations of object i_k, remove i_k from the set I, and replace k by $k - 1$. If $k = 0$ go to Step 5. Otherwise repeat Step 3.
4. If $k = m$ then replace C^* by B and store the objects and their corresponding locations as the best layout known so far, and go to Step 3. On the other hand if $k < m$, k is replaced by $k + 1$, and object i_k is placed in I. Place i_k into a vacant location and go to Step 2.

5. Stop. The optimal solution is C* and the corresponding layout is optimal.

Some Comments

a. Step 1 is an initialization step. In step 3 the current partial solution is fathomed because $B \geq C^*$. If object i_k could not be assigned anywhere then the level of the tree is reduced by 1 and the current assignment of object i_{k-1} is prohibited. In Step 4 the incumbent is updated, or else a new object is located.

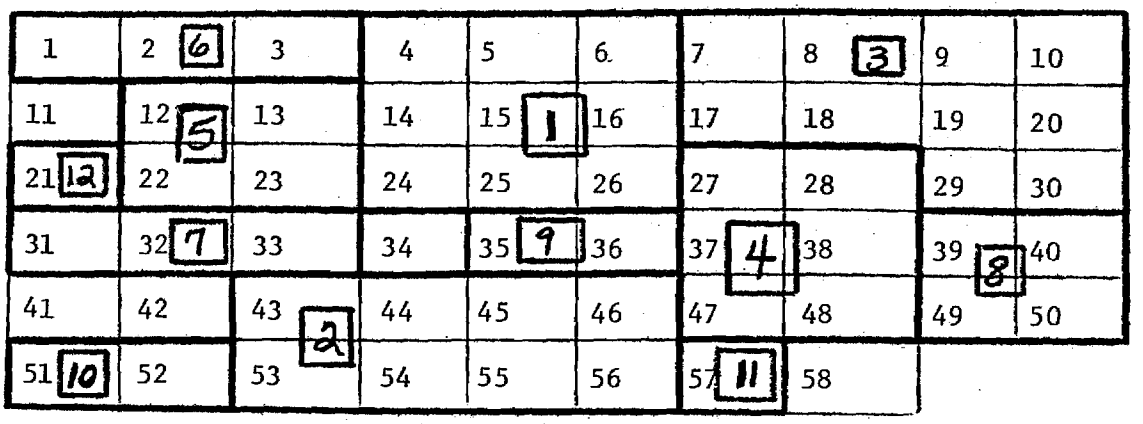
b. Given a partial assignment and a new object to be located, there are various methods to determine the location of this object. The location which minimizes the total weighted interaction with the already assigned objects is used here.

c. Many other rules for choosing a new object to be located could be used. For example, the object with maximum interaction with the already assigned objects may be chosen. A more complicated choice would be the object which if assigned optimally with respect to the already assigned objects, would provide the minimal lower bound of completion.

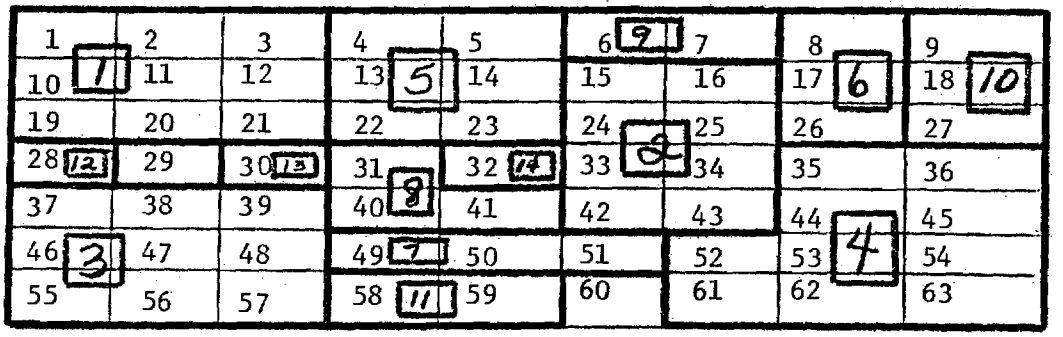
III.4 Computational Experience

In this section we will report our computational experience with two sample problems whose details are given later in the section. The branch and bound technique provided and verified the optimal solutions of both problems.

The first problem involved the assignment of 12 objects with a total area of 53 blocks to an area of 58 blocks. The optimal objective 14079 was obtained in 23.0 seconds on a Univac 1108, and is shown below.



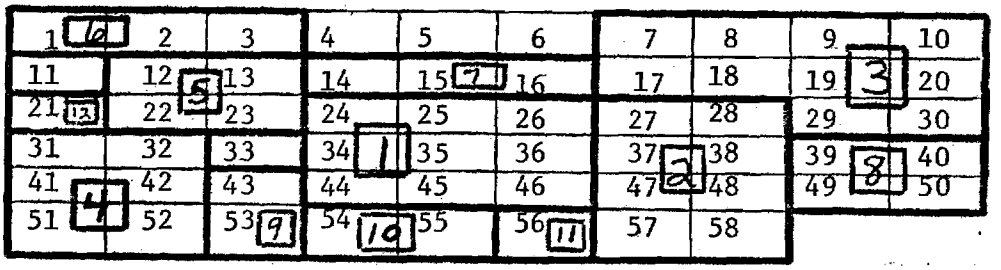
The second problem involved the assignment of 14 objects which occupy an area of 61 blocks, to an area of 63 blocks. The optimal solution with objective 8170.5 was obtained in 4.4 seconds on a Univac 1108 and is depicted below.



Needless to say, a post-optimality analysis is needed where other factors which have not been explicitly considered can be introduced, and where minor modifications of the layout are made. To illustrate, block 34 was vacant in the first problem. A decision must be made regarding what to do with this block, e. g., shifting object 7 such that it occupies blocks 32, 33, and 34; or enlarging object 9 so that it occupies blocks 34, 35, and 36, etc.

An alternative set of suboptimal layout designs can also be provided by the branch and bound scheme. Because there are usually some factors which cannot be introduced into the model, the layout engineer may prefer

a suboptimal design to the optimal design. A suboptimal design of the first problem with objective 14109 is obtained in 1.2 seconds and is depicted below. This layout, or another suboptimal layout, may be preferred by the layout engineer.



Problem Statements

The data for the two problems of this section are summarized below. The total interaction or flow between each two objects and the candidate locations of each object in terms of blocks are listed.

Problem 1

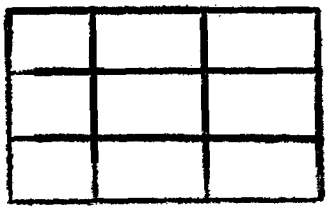
Interaction Matrix

	1	2	3	4	5	6	7	8	9	10	11	12
1	--	288	180	54	72	180	27	72	36	0	0	9
2		--	240	54	72	24	48	160	16	64	8	16
3			--	120	80	0	60	120	60	0	0	30
4				--	72	18	18	48	24	48	12	0
5					--	12	12	64	16	16	4	8
6						--	18	24	6	12	3	3
7							--	0	6	6	3	6
8								--	16	16	16	4
9									--	4	4	2
10										--	2	2
11											--	2
12												--

Candidate Locations

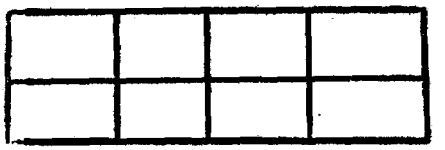
Object 1

1, 2, 3, 11, 12, 13, 21, 22, 23
24, 25, 26, 34, 35, 36, 44, 45, 46
28, 29, 30, 38, 39, 40, 48, 49, 50
31, 32, 33, 41, 42, 43, 51, 52, 53
4, 5, 6, 14, 15, 16, 24, 25, 26
27, 28, 29, 37, 38, 39, 47, 48, 49



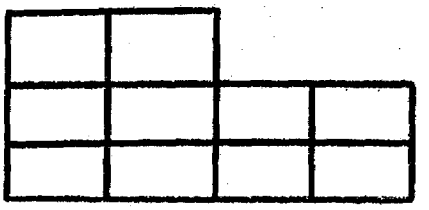
Object 2

31, 32, 33, 34, 41, 42, 43, 44
45, 46, 47, 48, 55, 56, 57, 58
19, 20, 29, 30, 39, 40, 49, 50
11, 12, 13, 14, 21, 22, 23, 24
43, 44, 45, 46, 53, 54, 55, 56
27, 37, 47, 57, 28, 38, 48, 58



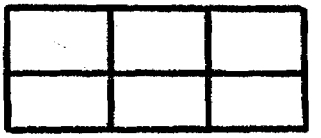
Object 3

7, 8, 9, 10, 17, 18, 19, 20, 29, 30
37, 38, 39, 40, 47, 48, 49, 50, 57, 58
5, 6, 15, 16, 17, 18, 25, 26, 27, 28
8, 9, 10, 18, 19, 20, 29, 30, 39, 40



Object 4

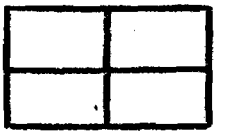
4, 14, 24, 5, 15, 25
1, 2, 3, 11, 12, 13
31, 32, 41, 42, 51, 52
14, 15, 16, 24, 25, 26
27, 28, 37, 38, 47, 48



Object 5

45, 46, 55, 56
39, 40, 49, 50
24, 25, 34, 35
22, 23, 12, 13

17, 18, 7, 8
4, 5, 14, 15
12, 13, 22, 23



Object 6

37, 47, 57
1, 2, 3
30, 40, 50
45, 46, 47

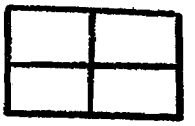
18, 19, 20
8, 9, 10
13, 23, 33

Object 7

48, 49, 50 10, 20, 30
31, 32, 33 14, 15, 16
29, 39, 49 16, 17, 18

Object 8

16, 17, 26, 27 19, 20, 29, 30
39, 40, 49, 50 33, 34, 43, 44
 6, 7, 16, 17 41, 42, 51, 52



Object 9

57, 58 51, 52 42, 43 8, 9
37, 38 9, 10 43, 53 35, 36
38, 48 25, 26 1, 11
29, 30 46, 47 13, 14
11, 12 41, 42 41, 31

Object 10

52, 53 7, 17 38, 39
51, 52 49,50 39, 40
54, 55 1, 2 42, 43

Object 11

any of 41, 42, 43, ..., 58

Object 12

any of 7, 8, 9, 10, 17, 18, 19, 21, 22, 32

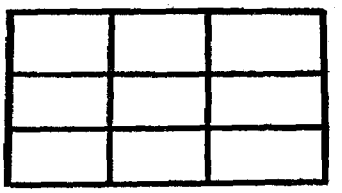
Problem 2

Interaction Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-	72	162	90	108	27	0	0	18	27	18	0	0	0
2		-	72	80	0	48	0	48	32	0	16	8	0	0
3			-	45	54	27	27	27	0	27	0	9	18	0
4				-	30	0	30	30	20	0	20	10	10	0
5					-	18	0	18	12	18	24	0	0	0
6						-	9	9	0	0	6	6	6	0
7							-	9	12	9	6	3	0	0
8								-	6	9	0	3	0	0
9									-	6	4	6	2	0
10										-	6	3	6	0
11											-	2	0	0
12												-	4	0
13													-	0
14														-

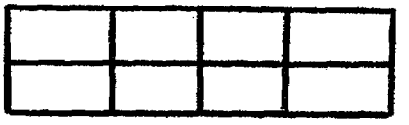
Object 1

1, 2, 3, 10, 11, 12, 19, 20, 21
14, 15, 16, 23, 24, 25, 32, 33, 34
37, 38, 39, 46, 47, 48, 55, 56, 57
7, 8, 9, 16, 17, 18, 25, 26, 27
40, 41, 42, 49, 50, 51, 58, 59, 60
6, 7, 8, 15, 16, 17, 24, 25, 26
30, 31, 32, 39, 40, 41, 48, 49, 50



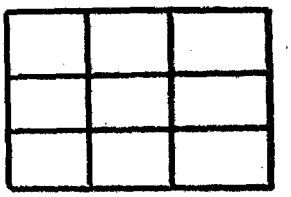
Object 2

13, 4, 22, 31, 5, 14, 23, 32
6, 7, 8, 9, 15, 16, 17, 18
28, 29, 37, 38, 46, 47, 55, 56
39, 40, 41, 42, 48, 49, 50, 51
35, 36, 44, 45, 53, 54, 62, 63
15, 16, 24, 25, 33, 34, 42, 43
22, 23, 24, 25, 31, 32, 33, 34



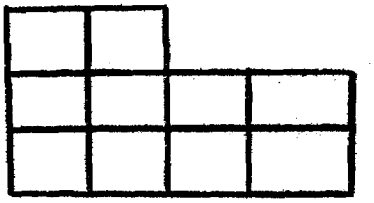
Object 3

43, 44, 45, 52, 53, 54, 61, 62, 63
37, 38, 39, 46, 47, 48, 55, 56, 57
13, 14, 15, 22, 23, 24, 31, 32, 33
39, 40, 41, 48, 49, 50, 57, 58, 59
7, 8, 9, 16, 17, 18, 25, 26, 27
33, 34, 35, 42, 43, 44, 51, 52, 53



Object 4

29, 30, 38, 39, 40, 41, 47, 48, 49, 50
35, 36, 44, 45, 52, 53, 54, 61, 62, 63
1, 2, 3, 4, 10, 11, 12, 13, 19, 20
21, 30, 4, 13, 22, 31, 5, 14, 23, 32
42, 51, 60, 43, 52, 61, 53, 62, 54, 63
6, 15, 24, 33, 7, 16, 25, 34, 36, 35

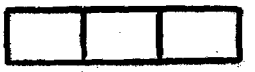


Object 5

15, 16, 17, 24, 25, 26
37, 38, 46, 47, 55, 56
1, 2, 3, 10, 11, 12
31, 32, 33, 40, 41, 42
6, 7, 15, 16, 24, 25
4, 13, 22, 5, 14, 23

Object 6

34, 35, 36 1, 10, 19
45, 54, 63 13, 14, 15
58, 59, 60 8, 17, 26
55, 56, 57

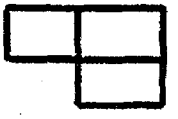


Object 7

42, 41, 60 49, 50, 51
4, 5, 6 3, 12, 21
9, 18, 27 38, 39, 40
40, 41, 42

Object 8

8, 9, 18 31, 40, 39
46, 55, 56 43, 44, 53
22, 23, 32 62, 63, 54
31, 40, 41



Object 9

6, 7 11, 12 50, 51
8, 9 1, 10 52, 61
34, 35 27, 36 62, 63

Object 10

57, 58, 59 3, 4, 5
45, 54, 63 5, 14, 23
7, 8, 9 34, 35, 36
9, 18, 27

Object 11

27, 36 22, 23
58, 59 20, 29
11, 20 19, 28
5, 6 40, 41

Object 12

any of 56, 9, 28, 31, 37, 63, 45, 39, 29

Object 13

any of 28, 46, 37, 56, 7, 24, 30, 19, 45

Object 14

any of 1, 2, ..., 63

REFERENCES

1. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables, Operations Research, Vol. 13, pp. 517-546, 1965.

2. Bazaraa, M. S., and J. J. Goode, "A Cutting Plane Algorithm for the Quadratic Set Covering Problem," Operations Research, to appear.

3. Bazaraa, M. S., R. L. Rardin, and C. M. Shetty, "Solving Quadratic Set Covering Problems by Implicit Enumeration," School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia.

4. Buffa, E. S., G. C. Armour, and T. E. Vollman, "Allocating Facilities with CRAFT," Harvard Business Review, Vol. 42, pp. 136-159, 1964.

5. Francis, R. L. and J. A. White, "Facility Layout and Location: An Analytical Approach," Prentice-Hall, 1974.

6. Geoffrion, A. M., "An Improved Implicit Enumeration Approach for Integer Programming," Operations Research, Vol. 17, 1969.

7. Lawler, E. L. and D. E. Wood, "Branch and Bound Methods: A Survey," Operations Research, Vol. 14, pp. 699-719, 1966.

8. Lee, R. C., and J. M. Moore, "CORELAP--Computerized Relationship Layout Planning," Journal of Industrial Engineering, Vol. 18, pp. 195-200, 1967.

9. Sheehof, J. M., and W. O. Evans, "Automatic Layout Design Program," The Journal of Industrial Engineering, Vol. 18, pp. 690-695, 1967.

10. Whitehead, N., and M. Z. Elders, "An Approach to the Optimum Layout of Single Story Buildings," Architect Journal, Vol. 139, pp. 1373-1380, 1964.

11. Zoller, K., and K. Adendorff, "Layout Planning by Computer Simulation," AIIE Transactions, Vol. 4, pp. 116-125, 1972.

IV. THE QUADRATIC ASSIGNMENT PROBLEM

In this section an exact branch and bound procedure which is able to produce optimal solutions for problems with twelve facilities or less, is developed. A new formulation of the problem and a duality based subgradient algorithm are pretested. Limited computational experience shows some difficulties with the latter approach relative to the size of the duality gap. Since the exact procedures were not satisfactory for large problems, a heuristic is developed which was able to produce good quality solutions with a reasonable computational effort. Computational experience with the exact and heuristic codes is presented.

IV.1 Introduction

Recall that the quadratic set covering problem P_1 can be formulated as follows:

$$\text{Minimize} \quad \sum_{\ell=1}^{n}\sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{i=1}^{m} c_{ijk\ell}x_{ij}x_{k\ell} + \sum_{j=1}^{n}\sum_{i=1}^{m} f_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad j = 1, 2, \ldots, n$$

$$x_{ij} \text{ is 0 or 1} \qquad i = 1, 2, \ldots, m \; j = 1, 2, \ldots, n$$

The following interpretation will be used here (see II.1 and II.2). We suppose that m objects are to be assigned to n locations ($n \geq m$). x_{ij} is 1 if object i is placed in location j and is 0 otherwise. $c_{ijk\ell}$ is the cost of mutually assigning object i to location j and object k to location ℓ. $c_{ijk\ell}$ is usually determined as the number of interactions u_{ik} between objects i and k weighted by the distance from location j to location ℓ, i. e., $c_{ijk\ell} = u_{ik}d_{j\ell}$. In some instances a fixed cost related to the mutual assignments x_{ij} and $x_{k\ell}$ may be incurred, say $\alpha_{ijk\ell}$ (See Graves and Whinston

[10]), in which case $c_{ijk\ell} = u_{ik}d_{j\ell} + \alpha_{ijk\ell}$. f_{ij} is the fixed cost of assigning object i to location j.

As discussed in II.2, applications of the problem range from location of indivisible facilities (Koopmans and Beckman [17]), building layout (Whitehead and Elders [26]), control panel layout (Dorris [5]), to wiring design in the placement of electronic components in assemblies (Breuer [3], Gaschutz and Ahrens [7], and Steinberg [24]).

Various procedures for solving the problem have been suggested in the literature, including both exact and heuristic methods. Exact methods include the integer programming formulation of (Lawler [19]) and several branch and bound procedures. The latter class of methods include the single assignment procedures of (Gilmore [9] and Lawler [17]) and the double assignment methods of (Land [18] and Gavett and Plyter [8]). These methods have either not been tested or else found impractical with n larger than 7 (See Pierce and Crowston [23]).

Several heuristics have been reported in the literature. The reader may refer to the procedures of (Armour and Buffa [1], Gaschutz and Ahrens [7], Gilmore [9], Graves and Whinston [10], Heider [12], Hillier [15], Hillier and Connors [16], Neghabat [20], Nugent, Vollman, and Ruml [21], Pegels [23], Steinberg [24], Whitehead and Elders [26], Vollman, Nugent and Zartler[25]). The reader may also refer to the papers of (Hannan and Kurtzberg[11]) and (Pierce and Crowston [23]), as well as the book of (Francis and White [6, Chapter 8]) which compare among various methods of solving the quadratic assignment problem.

In IV.2 we present an exact branch and bound procedure, which can be used to obtain optimal and suboptimal solutions with any apriori desired degree of accuracy. Of course the solution time increases as a function

of the accuracy desired. The procedure was able to optimally solve problems with size up to 12 facilities. The problem of adding new facilities to an existing layout is also discussed. In IV.3 a reformulation of the problem based on double assignment was devised. A duality based method is suggested to solve the new formulation. The approach did not prove particularly successful from a computational point of view for several reasons which are discussed. A heuristic procedure is reported in IV.4. Computational experience with both exact and heuristic codes is reported in IV.5.

IV.2 An Exact Branch and Bound Procedure

A branch and bound solution procedure which can be used to obtain optimal and "qualified" suboptimal solutions will be described here. The following notation will be used. The location to which object i is assigned is denoted by $\ell(i)$. Hence the mutual cost of assigning object i and j is $(u_{ij} + u_{ji})\, d_{\ell(i)\ell(j)}$ (assuming that $d_{\ell(i)\ell(j)} = d_{\ell(j)\ell(i)}$) and the fixed costs of these assignments (if any) are $f_{i\ell(i)} + f_{j\ell(j)}$.

Decision Tree and General Framework

At each stage of the algorithm, we have a set of objects that has already been assigned to certain locations. These already assigned objects form a partial solution of the assignment problem. In order to obtain a feasible solution, i. e., a complete assignment, we must find a completion of the partial solution. Rather than considering all the possible ways of completing the partial solution, we first investigate whether the partial solution may lead to a complete solution, with an objective value which is smaller than the best complete solution that we already have. This is done by calculating a lower bound on the cost of the partial solution plus the cost of completing this partial solution.

Let B be the lower bound and let C* be the cost of the best available assignment. If $B \geq C^*$ then any completion of the partial solution can lead to no improvement. If this is the case, the partial solution is said to be fathomed, and is abandoned. On the other hand, if $B < C^*$, it is worthwhile to pursue the partial solution, by seeking to assign more objects.

Calculation of the Lower Bound

Suppose that a set of objects in the set I have already been assigned to a set of locations J. In particular, suppose that object i is assigned to location $\ell(i)$. The cost of this partial solution, C_1, can be determined exactly as follows:

$$C_1 = \sum_{i \in I} f_{i\ell(i)} + \sum_{\substack{j \in I \\ j \neq i}} \sum_{i \in I} u_{ij} d_{\ell(i)\ell(j)}$$

The cost of completing this partial solution consists of the fixed costs of assigning objects to locations, plus two types of interaction costs. The interaction from unassigned objects to assigned objects, with bound C_2, plus the interaction between the unassigned objects themselves, with bound C_3.

C_2: bound on cost from unassigned objects to assigned objects.

C_2 is the optimal cost of the following linear assignment problem.

$$\text{Minimize} \quad \sum_{j \notin J} \sum_{i \notin I} b_{ij} x_{ij}$$

$$\text{Subject to} \quad \sum_{j \notin J} x_{ij} = 1 \quad \text{for each } i \notin I$$

$$\sum_{i \notin I} x_{ij} \leq 1 \quad \text{for each } j \notin J$$

$$x_{ij} \geq 0 \quad \text{for each } i \notin I,\ j \notin J$$

where b_{ij} is a bound on the cost resulting from assigning the unassigned object i to the unassigned location j, e. g.,

$$b_{ij} = f_{ij} + \sum_{t \in J} u_{it} d_{j\ell(t)} + \sum_{t \in J} u_{ti} d_{\ell(t)j}$$

Of course, a complete solution of the linear assignment problem can be replaced by the simpler task of reducing the matrix (b_{ij}) such that it has at least one zero in each row and each column by subtracting the minima of the rows from the rows, and the minima of the columns from the resulting columns. C_2 can then be replaced by the sum of the row minima plus the sum of the smallest $m - |I|$ column minima, where $|I|$ is the number of already assigned objects. Of course, this calculation is simpler, but the bound is not as tight as the bound obtained from solving the assignment problem. Both of these methods for calculating C_2 are used in the computational experience section.

C_3: bound on cost between unassigned objects and themselves.

Strategy 1

In order to calculate C_3, rank the interactions u_{ij} in a descending order for $i, j \notin I$, and rank the distances d_{ij} in an ascending order for $i, j \notin J$. This results in two vectors of size $(m - |I|)^2$ and $(n - |J|)^2$. If $n > m$, complete the size of the first vector such that it has $(n - |J|)^2$ entries by adding zeros to it from the bottom. Then C_3 is the inner product of the two vectors. In other words, C_3 is calculated by matching the largest interaction among unassigned elements to the smallest distance between unassigned locations, and 2nd largest interaction to 2nd smallest distance, and so forth. Clearly, this procedure will give a lower bound on the cost among unassigned elements and themselves.

Strategy 2

A bound on the cost of interaction among unassigned objects can be alternatively found by solving a linear assignment problem whose cost matrix is constructed as follows. For each unlocated element i, rank the interactions between it and all other unlocated elements in a descending order. Similarly, for each vacant location j, rank the distances between it and all other vacant locations in an ascending order. To calculate the entry e_{ij} find the inner product of the above two vectors. C_3 is found by solving the following linear assignment problem:

$$\text{Minimize} \quad \sum_{j \notin J} \sum_{i \notin I} e_{ij} x_{ij}$$

$$\text{Subject to} \quad \sum_{j \notin J} x_{ij} = 1 \quad \text{for each } i \notin I$$

$$\sum_{i \notin I} x_{ij} \leq 1 \quad \text{for each } j \notin J$$

$$x_{ij} \geq 0 \quad \text{for each } i \notin I,\ j \notin J$$

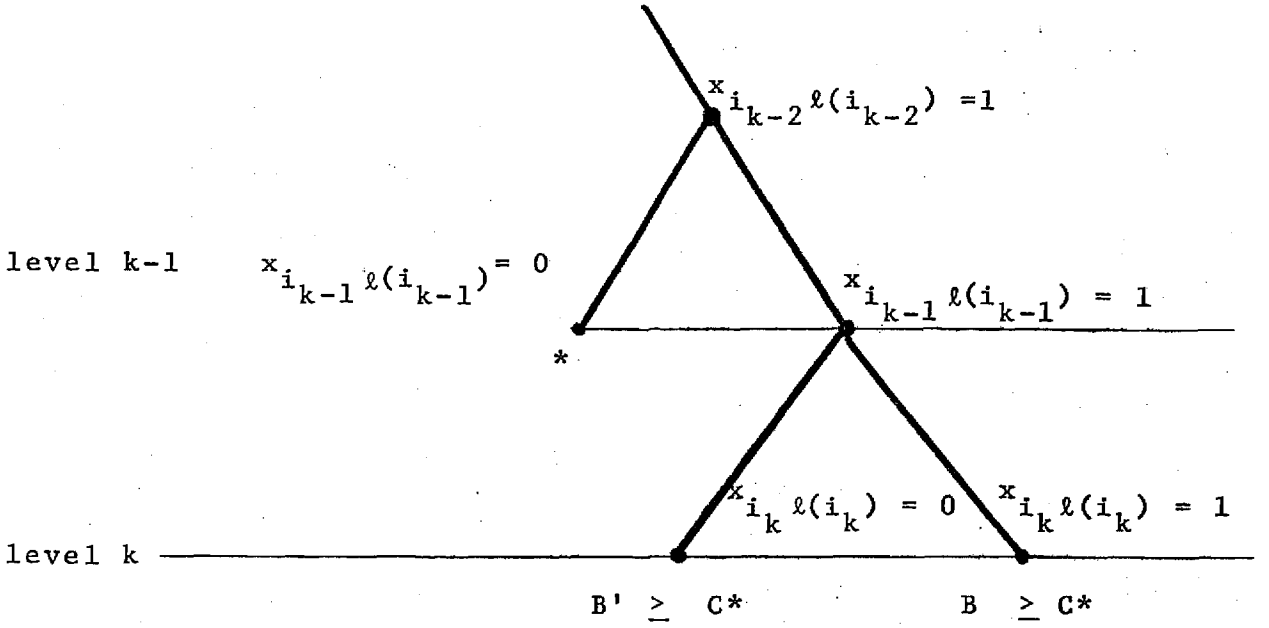
Needless to say, the above assignment problem can be combined with the assignment problem in the C_2 calculation to give $C_2 + C_3$. The overall lower bound $B = C_1 + C_2 + C_3$ is now calculated.

Continuation of the Search

Fathoming (Backward Move)

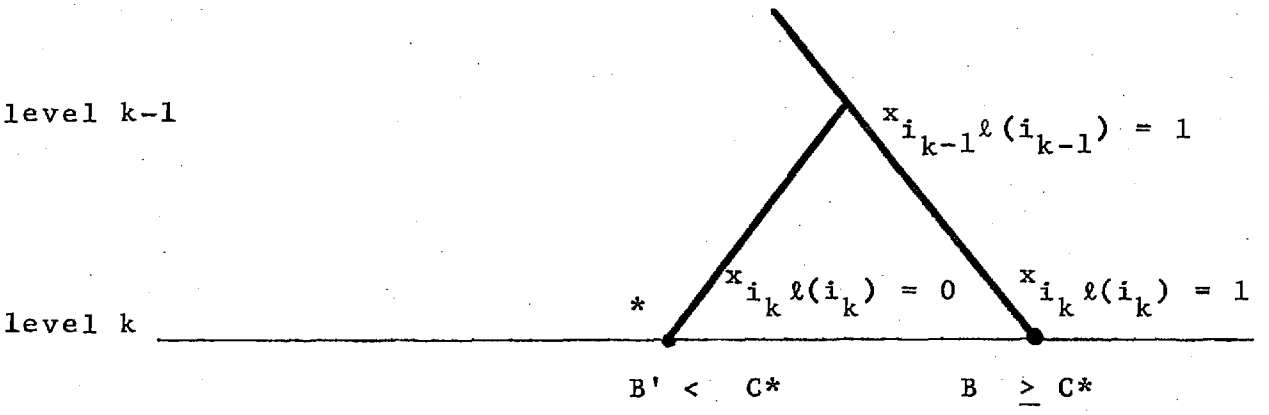
Suppose that k objects indexed by the set I have already been assigned to k locations indexed by the set J. The level of the trees is called k. A bound on the cost resulting from all completions of the current partial solution B is calculated as discussed above. If $B \geq C^*$, where C^* is the current best known cost of a complete assignment, then the partial solution is fathomed. The last assignment, i. e., the kth assignment is

banned, or prohibited, in the hope that this will lead to an improved completion. For example, if the kth assignment involved placing object i_k in location $\ell(i_k)$, then $x_{i_k\,\ell(i_k)}$ is forced to be zero. i_k is then placed in the unassigned list of objects, (i. e., i_k removed from I), and similarly $\ell(i_k)$ is removed from the unassigned list of locations (i. e., $\ell(i_k)$ is removed from J). A new bound B' is calculated, in exactly the same manner as explained above, except of course, that the assignment $x_{i_k\,\ell(i_k)} = 1$ is prohibited, say by forcing $b_{i_k\,\ell(i_k)} = +\infty$ while solving the linear assignment problem. If B' is still $\geq$ C*, then the partial solution of the first k-1 assignments, while banning the assignment i_k to $\ell(i_k)$, can still lead to no improved solutions. Since the first k-1 assignments with $x_{i_k\,\ell(i_k)} = 1$ and $x_{i_k\,\ell(i_k)} = 0$ lead to no improvement, then all the possibilities at level k have been exhausted, and prohibiting the assignment at level k-1 is now possible. This is called strong fathoming, and the level of the tree is thus reduced by 1 unit, by prohibiting the assignment at level k-1, and the process is repeated. If on the other hand, the bound B' is less that C*, which will be referred to as weak fathoming, then object i_k is assigned to some other unassigned location. This is discussed in more detail in the forward move of the search. The cases of strong and weak fathoming are depicted below.



Strong Fathoming

* Continue at level k-1 by trying to assign object i_{k-1} to some unassigned location $\neq \ell_{(i_{k-1})}$.



Weak Fathoming

* Continue at level k by trying to assign object i_k to some unassigned location $\neq \ell(i_k)$.

Progress (Forward Move)

If $B < C^*$ we must choose an object i_{k+1} for assignment. For example, we may choose i_{k+1} to be an unassigned object with maximum interactions with already assigned objects, or choose i_{k+1} to be an unassigned object with maximum interactions with the most recently assigned object i_k. This object is assigned to an unassigned location $\ell(i_{k+1})$ which is not prohibited. This location $\ell(i_{k+1})$ can be picked in such a way that the total weighted interaction with assigned objects is minimal, e. g., choose $\ell(i_{k+1})$ which Minimizes $\sum_{j=1}^{k} u_{i_{k+1}i_j} d_{t\ell(i_j)} + \sum_{j=1}^{k} u_{i_j i_{k+1}} d_{\ell(i_j)t}$ over $t \notin J$ such that $x_{i_{k+1}t} = 1$ is not prohibited.

Choosing an object i_{k+1} and a location $\ell(i_{k+1})$ in the above fashion, increases the level of the tree by 1. A bound on completing the partial solution is calculated as before and the process is repeated.

Needless to say, when the level of the tree is m, and if the cost is less than C*, then C* is replaced by the cost, and the corresponding assignment is stored.

Termination

We have described forward and backward progress of the tree search. If the level of the tree ever reaches value zero, then we stop. This would mean that we are currently at level one, and are trying to backtrack. This means that all possible assignments under $x_{i_1\ell(i_1)} = 1$ and $x_{i_1\ell(i_1)} = 0$ have already been enumerated, i. e., all possible ways of assigning the m objects are enumerated, and we stop. The stored assignment and corresponding C* give the optimal solution.

Summary of the Algorithm

We have discussed above all the details required to describe the following

solution procedure of the quadratic assignment problem.

Initialization Step

Let $P(i) = \emptyset$ for $i = 1,2,\ldots,m$ (prohibited locations). Let $C^* = \infty$. Choose an object i_1 and place it in location $\ell(i_1)$. i_1 can be determined by maximizing $\sum_{j=1}^{m} u_{ij}$ for $i = 1,2,\ldots,m$, and $\ell(i_1)$ can be determined by minimizing $\sum_{i=1}^{n} d_{ij}$ for $j = 1,2,\ldots,n$. Let $I = \{i_1\}$ and $J = \{j_1\}$. Let $k = 1$ and go to step 1.

Step 1 (Forward Move)

Calculate a lower bound B on all completions of current partial solution. $B = C_1 + C_2 + C_3$, where C_1, C_2, and C_3 are calculated as shown earlier, with the exception that $b_{ij} = \infty$ if $j \in P(i)$. If $B \geq C^*$ go to step 2.

Otherwise pick $i_{k+1} \notin I$ such that $u_{i_{k+1}i_k} = \underset{i \notin I}{\text{Max}}\ u_{ii_k}$, and place i_{k+1} in $\ell(i_{k+1})$ where $\ell(i_{k+1})$ is determined by

$$\text{Minimizing}_{\substack{j \notin J \\ j \notin P(i_{k+1})}} \quad f_{i_{k+1}j} + \sum_{t=1}^{k} u_{i_{k+1}i_t} d_{j\ell(i_t)} + \sum_{t=1}^{k} u_{i_t i_{k+1}} d_{\ell(i_t)j}$$

Replace I by $I \cup \{i_{k+1}\}$ and J by $J \cup \{\ell(i_{k+1})\}$.

If $k = m - 1$, calculate the new value of C_1 (cost of complete assignment). If $C_1 < C^*$, replace C^* by C_1, store $(i_t, \ell_{(i_t)})$ for $t = 1,2,\ldots,m$, replace k by m and go to step 2. If $C_1 \geq C^*$ then replace k by m and go to step 2. If $k < m - 1$, then replace k by $k + 1$ and repeat step 1.

Step 2 (Fathom)

i_k is removed from I and $\ell(i_k)$ is removed from J. Replace $P(i_k)$ by $P(i_k) \cup \{\ell(i_k)\}$. Calculate a lower bound B on completions of the partial solution $x_{i_t\ell(i_t)} = 1$ for $t = 1,2,\ldots,k-1$ and $x_{i_k\ell(i_k)} = 0$, where $B = C_1 + C_2 + C_3$, and $b_{ij} = +\infty$ if $j \in P(i)$. If $B \geq C^*$ go to step 3. Otherwise assign i_k to a location $t \notin J \cup P(i_k)$ such that the cost

$f_{i_k t} + \sum_{j=1}^{k-1} u_{i_k i_j} d_{t\ell(i_j)} + \sum_{j=1}^{k-1} u_{i_j i_k} d_{\ell(i_j)t}$ is minimized over $t \notin J \cup P(i_k)$. i_k is added to I, and t, the new $\ell(i_k)$, is added to J. Go to step 1.

Step 3 (Strong Fathom)

i_{k-1} is deleted from I and $\ell(i_{k-1})$ is deleted from J. $\ell(i_{k-1})$ is placed in $P(i_{k-1})$ and $P(i_k)$ is replaced by the empty set. k is replaced by k-1. If k = 0 go to step 4, otherwise go to step 2.

Step 4 (Termination)

Search has been completed. Optimal cost is C* and its corresponding assignment $(i_k, \ell(i_k))$, k = 1,2,...,m is the optimal assignment. Stop.

The following points are helpful to keep in mind.

1. During the initialization step an upper bound C* = ∞ is used. As the search progresses, C* denotes the objective of the best available complete assignment. Note that P(i) represents the locations that object i cannot be assigned to. These are initialized by the empty sets.
2. Step 1 represents a forward step, where the level of the tree increases by 1 unit. In this case the bound is less than C*, and hence a complete solution with an objective better than C* is possible.
3. Step 2 is a fathoming step, where $B \geq C^*$. In this case the last assignment is prohibited. Immediately a new bound is calculated. If the new bound is less than C*, then a forward move is made. But if the bound is still greater than or equal to C*, then a stong fathoming (Step 3) is made, and the level of the tree is reduced. Of course, strong fathoming is most desirable, since it

avoids the expensive task of trying to locate object i_k to a free location other than $\ell(i_k)$.

Sub-Optimal Solutions

Due to the highly combinatorial nature of the problem, the task of finding and verifying the optimal solution within a reasonable computational time, for problems with $n > 15$, is almost impossible. In order to solve larger problems, we must resort to sub-optimal solutions. The branch and bound procedure itself can be used to obtain sub-optimal solutions. We propose two methods for implementing this general strategy.

Method 1

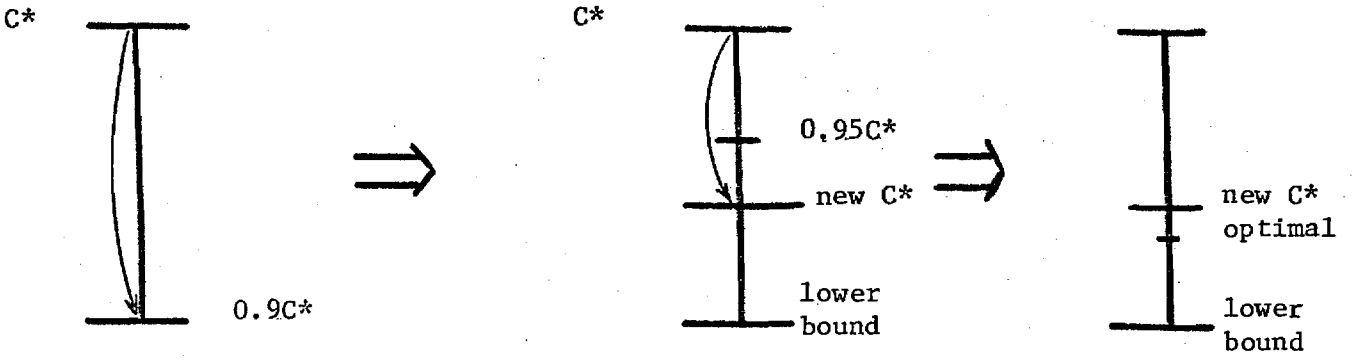
Recall that a partial solution is fathomed, if the lower bound on all completions is at least as big as C*, the best known objective. Suppose that a partial solution is fathomed if $B \geq \alpha C^*$, where $\alpha \in [0,1]$. In this case, the partial solution is abandoned if there is no hope that it leads to an objective which is better than αC^*. The objective of this simple strategy is clear. We want to fathom, i. e., abandon partial solutions, quickly, even if they lead to a slight improvement. Of course, as a new C* is found, then we fathom whenever the bound is greater or equal to α times the new C*. The procedure continues until we cannot find a feasible solution with an objective less than αC^*. So we have a feasible assignment with objective C*, coupled with the statement that the optimal objective is greater than or equal to αC^*.

Choice of α

Of course, if α is small, then fathoming will speed up considerably, resulting in a small computational effort. But on the other hand, the quality of the best obtained feasible solution is not satisfactory. We recommend values of $\alpha \geq 0.9$ depending on the accuracy required.

Exact Solution by Stepped Fathoming

A slight modification of the above scheme can be used to find the exact optimal solution, and still reducing the portion of the tree explicitly enumerated. Suppose that an initial α_1 is used, say $\alpha_1 = 0.90$. Eventually we achieve a final C* and cannot find a feasible solution with objective less than 0.90C*, i. e., 0.9C* is a lower bound on all solutions. We either stop here, or else switch to $\alpha_2 \geq \alpha_1$, say $\alpha_2 = 0.95$. We start the search from the stored best solution corresponding to C*, which also included all the prohibited locations. We are either able to find a feasible solution with objective less than 0.95C* (none less than 0.9C* exist), and repeat the process, or else conclude that none exists. We may then switch to 97% fathoming and eventually to 100% fathoming. This, of course, will lead to the optimal solution. Even though parts of the tree will be searched more than once, the quick fathoming outweighs any repeated effort.



No solution with Objective < 0.9C* exists

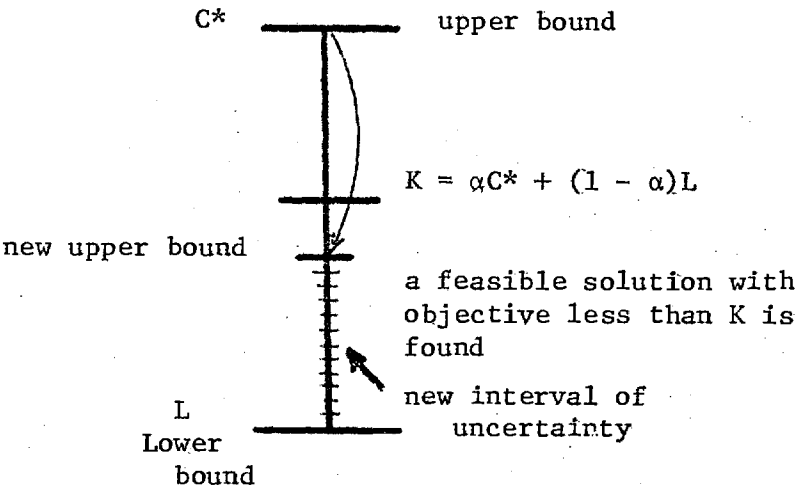
A solution with objective < 0.95C* found

100% fathoming

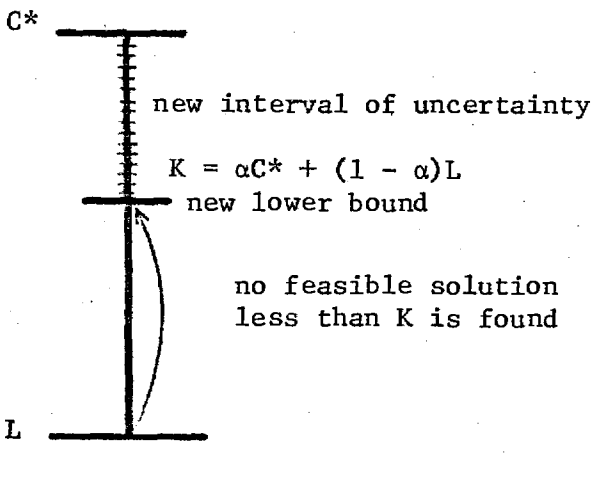
Method 2

At each stage of the algorithm, we have an upper bound C*. A lower bound on the overall problem L, can be devised. Rather than fathoming on C*, suppose we fathom on $K = \alpha C^* + (1 - \alpha)L$, where $\alpha \in [0,1]$. Since

$L < C^*$, then $K = \alpha C^* + (1 - \alpha)L \leq C^*$. Two cases are possible. In the first case, we will be able to find a complete solution with objective less than K. The objective of this new solution becomes the new upper bound C^* and the process is repeated. In the second case, we will not be able to find such a solution. This automatically implies that there are no solutions with objective less than K, and hence K itself is the new lower bound. The process is repeated. From this we keep narrowing the gap between the lower and upper bounds, either by lowering the upper bound (finding an improved feasible assignment) or by raising the lower bound (by finding that there is no feasible solution with objective lower than the current lower bound). When the difference between the lower and upper bounds is smaller than a prescribed tolerance, we either stop, or switch to 100% fathoming. Of course, if the tolerance is zero, or if we switch to 100% fathoming (i. e., switch α to 1), then we will find the true optimal solution.



Upper bound reduced



Lower bound raised

Choice of α

α is any number in the interval [0,1]. Of course, if α is close to 1, then we are in effect fathoming on a number very close to C^*, and the search

will not speed considerably. On the other hand, if α is close to zero, then improvement is achieved only if we obtain a feasible solution very close to the overall lower bound. In this case, fathoming will be fast, but it is likely not to obtain feasible solutions less than K. So again a tradeoff is required here.

Calculation of initial overall lower and upper bounds

Initial lower and upper bounds are needed for implementing the above fathoming scheme. To calculate the lower bound, first calculate a lower bound b_{ij} on the cost of assigning object i to location j, for i = 1,2,...,m and j = 1,2,...,n. This is done as discussed earlier. Then a linear assignment problem is solved to find L. More precisely, let L be the optimal objective of the following problem.

$$\text{Minimize} \quad \sum_{j=1}^{n} \sum_{i=1}^{m} b_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,2,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad j = 1,2,\ldots,n$$

$$x_{ij} \geq 0 \qquad \text{all } i \text{ and } j$$

An upper bound C* is immediately available, by calculating the quadratic cost of the optimal assignment resulting from the above problem. Now Method 2 can be initiated.

Adding New Facilities to an Existing Layout

In many applications a large number of objects is already preassigned, and only some new objects are to be placed, such that the overall cost is minimized. In this case, the above algorithm can be applied with few obvious modifications in the calculations. Since the preassigned objects and their locations will always be assigned, then these objects will always be in the set I and their locations will always be in the set J. In the tree search,

if γ objects are already assigned, we start the search by assigning more objects, i. e., the level of the tree starts at $\gamma + 1$. If the level of the tree ever becomes γ, then we stop. If an overall lower bound on all feasible solutions to the problem is desired, we recommend calculating $C_1 + C_2 + C_3$ as before, where I and J are the indices of the preassigned objects and their locations.

It is interesting to note that the problem of assigning m facilities anew is usually harder to solve than a similar problem where m new facilities are to be assigned to an existing layout. This is due to the fact that the lower bounds in the latter case are usually tighter, thus allowing quicker fathoming.

IV.3. Reformulation of the Problem

At a certain stage, an alternative formulation for the quadratic assignment problem is sought in such a way that an approach based on solving the dual problem would be involved. Under all outcomes, it would either produce a solution to the original problem or at least a set of dual variables which, when appropriately added in the original cost data would result in higher bounds at the stage of applying the tree search scheme. This would be expected to make search faster and hence reduce the computational difficulty of the problem in its known form.

Reformulation

Define the following:

$$c_{ijk\ell} = f_{ij} + f_{k\ell} + (u_{ik} + u_{ki})\, d_{j\ell}$$

$$y_{ijk\ell} = \begin{cases} 1 & \text{if object i is located in j and object k is located in } \ell \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if object i is located in j} \\ 0 & \text{otherwise} \end{cases}$$

α_i is the number of objects interacting with object i.

Then the problem could be reformulated as follows:

P2: Minimize $\Sigma_i \Sigma_j \Sigma_k \Sigma_\ell \; c_{ijk\ell} y_{ijk\ell}$

s.t.

			dual variables
(a)	$\Sigma_k \Sigma_\ell \; y_{ijk\ell} - \alpha_i x_{ij} = 0$	for all i,j	u_{ij}
	$\Sigma_i \Sigma_j \; y_{ijk\ell} - \alpha_k x_{k\ell} = 0$	for all k,ℓ	$v_{k\ell}$
(b)	$\Sigma_i \Sigma_k \; y_{ijk\ell} = 1$	for all j,ℓ	
(c)	$\Sigma_i \; x_{ij} \leq 1$	for all j	
	$\Sigma_j \; x_{ij} = 1$	for all i	
(d)	$y_{ijk\ell}$ and x_{ij} is 0 or 1	for all i,j,k,ℓ	

The above formulation can be thought of as a pair assignment formulation where the number of variables significantly increase. Note that the constraints are linear, except for, integrality of the variables. The formulation implies that $y_{ijk\ell} = 1$ if and only if $x_{ij} = x_{k\ell} = 1$. The constraint set (a) specifies that the number of nonzero $y_{ijk\ell}$ has to comply with the number of actual connections to element i(k) and the fact that it is in location j(ℓ) or not. The set (b) specifies that $y_{ijk\ell}$ has to comply with the fact that one element only can be in a location. Constraint set (c) is the standard assignment problem constraint set (i. e., each location can have at most one element and each element can occupy only one location). The 0-1 condition is expressed in the set (d).

Problem P2 can be rewritten as follows, where for the purpose of simplification of notation n is assumed equal to m, the dual vectors u and v are combined as w, and α is a diagonal matrix with obvious entries.

			Dual vector
P3:	Minimize	cy	
	s.t.	$Ay - \alpha x = 0$	w
		$By = 1$	
		$Dx = 1$	

x,y are vectors of 0-1 variables

A dual of P3 can be devised by incorporating the first set of constraints into the objective function, leading to problem P4.

P4: Maximize $\theta(w)$

w is a vector of unrestricted variables

where $\theta(w) =$ Minimum $\{(c + wA)y - w \alpha x\}$

s.t. $By = 1$

$Dx = 1$

x,y are vectors of 0-1 variables

which could be rewritten as

$$\theta(w) = \underbrace{\left\{\begin{array}{ll}\text{Minimum} & (c + wA)y \\ \text{s.t.} & By = 1 \\ & y \text{ is a vector of 0-1 variables}\end{array}\right\}}_{\text{I}} + \underbrace{\left\{\begin{array}{ll}\text{Minimum} & -w \alpha x \\ & Dx = 1 \\ & x \text{ is a vector of 0-1 variables}\end{array}\right\}}_{\text{II}}$$

alternatively, I and II could be written in the forms:

$$\text{I:} \quad \text{Minimum} \quad \Sigma_i \Sigma_j \Sigma_k \Sigma_\ell (c_{ijk\ell} + u_{ij} + v_{k\ell})\, y_{ijk\ell}$$

$$\text{s.t.} \quad \Sigma_i \Sigma_k \; y_{ijk\ell} = 1 \text{ for all } j,\ell$$

$y_{ijk\ell}$ is 0 or 1 for each i,j,k,ℓ

II. Minimum $\Sigma_i -\alpha_i \Sigma_j (u_{ij} + v_{ij})x_{ij}$

s.t. $\Sigma_i x_{ij} = 1$ for each j

$\Sigma_j x_{ij} = 1$ for each i

x_{ij} is 0 or 1 for each i,j

It is obvious that for a known vector w, whose components are the u_{ij}'s and the $v_{k\ell}$'s, part I can be evaluated by solving n(n - 1) trival knapsack problems as follows: for any given pair j and ℓ examine ($c_{ijk\ell}$ + u_{ij} + $v_{k\ell}$) over all i and k. Let $y_{ijk\ell} = 0$ for all except the one with the smallest $c_{ijk\ell} + u_{ij} + v_{k\ell}$, which has a value of 1. For part II, once w is known, its value can be obtained by solving a linear assignment problem.

The features of this formulation give rise to a simple procedure for solving the dual problem. Now we discuss the steps of this procedure.

Steps of the Solution Procedure, the Ascent Scheme

1) Choose starting values for the components of the w vector
2) Set μ = Max $\Theta(w) = -\infty$, ν = best quadratic objective function = $+\infty$
3) Evaluate $\Theta(w)$ using the current w vector, update μ as appropriate
4) Consider the primal solution obtained from evaluating II in Step 2, evaluate its quadratic cost. Update ν as appropriate.
5) Compare μ and ν, if they are close enough go to Step 9, otherwise continue.
6) Update the components of the w vector as follows: $w \leftarrow w + \lambda(Ay - \alpha x)$,

   i. e., $u_{ij} \leftarrow u_{ij} + \lambda(\Sigma_k \Sigma_\ell y_{ijk\ell} - \alpha_i x_{ij})$

   $v_{k\ell} \leftarrow v_{k\ell} + \lambda(\Sigma_i \Sigma_j y_{ijk\ell} - \alpha_k x_{k\ell})$

   where $\lambda > 0$ is a suitable stepsize, the choice of which will be discussed later.

7) Check whether the new components of the w vector are exactly the same as in the preceeding iteration, if so this means that:

$$\Sigma_k \Sigma_\ell y_{ijk\ell} - \alpha_i x_{ij} = 0 \text{ for each } i,j$$

$$\Sigma_i \Sigma_j y_{ijk\ell} - \alpha_k x_{k\ell} = 0 \text{ for each } k,\ell$$

i. e., both primal feasibility and dual maximum are achieved. If this occurs, we stop and the solution in hand is the optimal solution to the problem. Otherwise continue.

8) Check whether a considerable number of iterations was conducted without improving μ. If so proceed to Step 9, otherwise return to Step 3.

9) Terminate.

The above procedure is a subgradient optimization technique for maximizing the piecewise linear dual objective function Θ. Note that moving along $Ay - \alpha x$ a small step will either improve Θ or at least get us closer to the optimal point. For motivation, convergence, and geometrical interpretation of general subgradient optimization procedure, the reader may refer to (Held and Karp [13], Held, Wolfe, and Crowder [14], and Bazaraa and Goode [2]).

As a result of applying this procedure either of the following three outcomes occur:

a. During the solution of the dual we get to a point where primal feasibility is achieved.

b. During the solution of the dual, we get a feasible primal solution, via the x_{ij}'s, whose quadratic cost is close enough to the max $\Theta(w)$ obtained so far.

c. Neither (a) nor (b) occurs and we cycle for a good number of times without a significant improvement.

If case c occurs, then we have to resort to a tree search procedure to solve the problem. However, the cost data ($c_{ijk\ell}$'s) are augmented by the values of the dual variables (u_{ij}'s and $v_{k\ell}$'s) obtained as a result of applying this procedure. This is likely to precondition the data, hence strengthen the calculated bounds and would speed up the search. In the sequel we discuss in some detail the outline of the tree search procedure.

The Tree Search Scheme

Suppose that at any general stage of the search we have some x_{ij}'s which are fixed at value 1. This automatically fixes some values of $y_{ijk\ell}$ at value 1 and some others at value 0. In particular, if $x_{ij} = 1$, and $x_{k\ell} = 1$, then $y_{ijk\ell} = 1$ and $y_{k\ell ij} = 1$. We find the cost of this partial solution by calculating

$$\sum_i \sum_j \sum_k \sum_\ell (c_{ijk\ell} + u_{ij} + v_{k\ell})y_{ijk\ell} + \Sigma_i \alpha_i \Sigma_j \quad (u_{ij} + v_{ij})x_{ij}$$

Now, we find a bound on the best completion of this solution, i. e.,

$$\left\{\begin{array}{l} \text{Minimize} \quad (c + wA)y \\ \text{s.t.} \qquad\quad B\,y = 1 \\ y \text{ is a vector of 0-1 variables. Some y's are fixed.} \end{array}\right\} \quad + \quad \left\{\begin{array}{l} \text{Minimize} \quad -\alpha\, w\, x \\ \text{s.t.} \qquad\quad D\,x = 1 \\ x \text{ is a vector of 0-1 variables. Some x's are fixed.} \end{array}\right\}$$

If the partial cost plus the bound on completing it is greater than or equal to the best known upper bound, the search along the current branch is terminated. Otherwise continue by trying to fix another x_{ij}. As a result of calculating the lower bound we get a feasible assignment and we calculate its quadratic cost. If the quadratic cost is less than the best known solution obtained so far, it is stored as the incumbent. Apart from these details, the general scheme of the tree search procedure would not differ much from the one given in Section IV.2.

Choice of Initial Dual Vector w and Stepsize λ

In this study an initial value w = 0 was chosen. Other possibilities exist, e. g., letting $u_{ij} = v_{ij} = -\beta_{ij}$, where β_{ij} in the dual variable corresponding to the nonnegativity constraint of x_{ij} while solving the linear assignment problem with matrix (b_{ij}).

As usually is the case with subgradient optimization techniques, a small stepsize, or a decreasing sequence of stepsizes must be used. One method that has also been tried is to normalize the subgradient vector $Ay - \alpha x$ (by dividing by its norm) and then moving a distance λ. A sequence of decreasing λ's, say (1.0, 0.4, 0.1) may be used where we switch to a smaller value of λ if a certain number of trials without increasing Θ is encountered.

Since subgradient optimization methods will almost certainly encounter zigzagging (see Bazaraa and Goode [2]), taking convex combinations of the directions at two successive iterations was employed. This indeed helped reduce the zigzagging phenomena.

Limited Experience with the Algorithm

The algorithm discussed above was tried on the 5 x 5 and 6 x 6 problems of (Nugent et. al., [21]). The 5 x 5 problem has an optimal objective of 25. The maximum value of Θ obtained is 22. In the course of dual optimization, however, the optimal solution with value 25 was obtained (Step 4). For the 6 x 6 problem whose optimal solution is 43, the ascent scheme was able to only achieve a dual objective of 36, and the optimal primal assignment with cost 43 was also obtained.

Even though the optimal solution of both problems that we tried were obtained, the lack of improvement of the dual function beyond the values of 22 and 36 was disappointing. This may be attributed to several factors.

First of all, the formulation was designed for the general nonsymmetric problem and hence did not make use of the symmetry of the problems that we tried. Secondly, the maximum of the dual objective only gives the solution of the continuous linear program of problem P2. This is due to the fact that integrality of x and y is a byproduct of the formulation after constraints (a) is placed into the objective function. Hence a duality gap is generally expected between problems P2 (or original problem) and P4. Due to the special structure of the problem, we have hoped that the duality gap (if any) is small. At this stage, we cannot confirm whether the difference between the dual objective obtained and the true optimal solution is due to a duality gap, inability to find the true maximum of the dual, or to both.

IV.4 A Heuristic Procedure

As will be detailed later in the computational experience station, the exact procedure was not able to find optimal solutions for large problems. A heuristic procedure is described below for obtaining "good" solutions in a reasonable computational time for larger problems.

Obtaining an Initial Solution

1. Rank the locations as follows:
   - i) Calculate R_i, the sum of distances from location i to all other locations.
   - ii) Rank the locations in an ascending order of R_i.
2. Rank the facilities as follows:
   - i) Calculate E^1_i, the number of facilities connected to facility i.
   - ii) Calculate E^2_i, the total interactions to and from facility i.
   - iii) Rank the facilities in a descending order of E^1_i.

iv) Rank the facilities in a descending order of E^2_i.

v) Find the combined index L_i for each facility, according to the rankings in iii) and iv).

vi) Rank the facilities in an ascending order of L_i.

3. At any general step k, check whether any facility remains unassigned, if so, proceed to Step 4, otherwise go to Step 7.

4. Pick the next element in the rank, locate it to the first available location in the rank, set key = 1, and proceed to Step .

5. Pick the element with maximum number of interactions with the most recently located facility, if none are available, go to Step 7, otherwise locate it so as to result in a minimal increase in the total cost. Set key = 2 and proceed to Step 6.

6. Exchange the element which has just been located with each of the previously located elements in turn, each time calculating the resulting change in cost. Maintain the partial assignment which causes the maximum reduction, if any; finally replace the original assignment with that one. If key = 1, return to Step 5, otherwise return to Step 3.

7. Terminate as all elements have been assigned.

Improving the Initial Solution

Once a complete assignment is available, it is possible to seek improvements by interchanging the locations of two facilities. Once an improvement is achieved, it is possible to either repeat the process in search for more improvements or stop. The process can be repeated until we get to the stage where we find that no further improvements could be achieved by performing single exchanges on the pattern in hand. At each stage, it is possible to either choose the first exchange which will result in an improvement or to choose the one which yields the maximum improvements.

We have chosen the strategy of testing the effect of interchanging each facility, with each other facility, storing the best resulting pattern and interchanging the two facilities if the best pattern provides a reduction in the total cost, otherwise the same process is repeated for the next facility and so on. At any stage, if two facilities are eventually interchanged, we start all over again. However, if we fail to achieve any further improvement by interchanging the location of two facilities in the best assignment, we take the next best assignment, chosen from amongst all the assignments resulting from the test of interchanges at the immediately preceeding step, and repeat the whole procedure.

The termination criterion is simple; we can either stop when the difference between the value of the best solution so far and the lower bound is within a prespecified limit or when we have repeated the overall procedure, each time starting with a different assignment, a number of times without improvement.

IV.5 Computational Experience

The experience gained with the solution procedures was in relation to the eight problems of (Nugent, et. al., [21]). First, we will discuss some details pertinent to the purely computational aspects of the exact procedure.

a) Choice of α

If a small value of α is chosen, the fictitious upper bounds tend to be tighter and hence the search becomes faster. However, we may increase the number of restarts this way. On the other hand, if α is large, then fathoming becomes weaker and the search tends to be lengthier but with a less number of restarts. The tradeoff is only computational and is data dependent.

During the course of our study, we noticed that due to the features of the search procedure, many successive good solutions are obtained very early

in the search and the optimum follows suit. As a result, we have adopted the strategy of choosing a small value of α at the beginning of the search and up to a certain stage where we switch to 100% fathoming. This is conducted by specifying an initial value of α and also specifying a difference between the actual upper and lower bounds at the achievement of which α is switched to 100%. If the difference is appropriately chosen, we will get to the stage where the upper bound is tight enough to speed up the search and also we will not have to restart the search, once the tree is enumerated, as there is no interval of uncertainty in this case.

We found that the choice of α = 70%, as a starting value, was adequate for all the problems solved. The difference between the two bounds after which we switch to 100% fathoming varied from one problem to another.

b) Solving the Linear Assignment Problem

As was mentioned in IV.2, the lower bound can be calculated by different methods. One procedure involves a linear assignment problem (LAP) to obtain a tighter lower bound. There is no need, however, to solve a fresh LAP each time a lower bound is to be calculated. It is possible to take any previous solution and update it according to the new cost matrix. Consequently, each optimal solution to the LAP under a certain cost matrix could be taken as a starting feasible solution the next time a need arises to solve a LAP under a different cost matrix. Several existing codes were incorporated in a code for the exact procedure and the solution times were compared. Finally, we have selected the code of McGinnis* to be used and the reported computational

*We would like to thank Mr. Leon F. McGinnis (N. C. State University) for providing us with his code which is based on the method of Glover, Klingman, and Kearny.

experience reflects its efficiency. For example, with problem 4006 (Table IV.1) the solution of the LAP was updated 26,367 times, while the total CPU time for the whole problem was about 490 seconds, i. e., under all circumstances, each updating process could not have taken more than 0.02 seconds.

Computational Experience

In the sequel, we summarize our experience with both the exact and the heuristic procedures.

1) The Exact Procedure

Tables IV.1 and IV.2 summarize the experience with the following two codes:

QAP3: a code for an algorithm based on calculating the lower bound $C_1 + C_2 + C_3$ by matching the vectors.

QAP7: a code for an algorithm based on calculating the lower bound $C_1 + C_2 + C_3$ by the reduction method augmented by solving a linear assignment problem only when the lower bound calculated by reduction is not higher than the current upper bound. Some of the statistics listed in these tables are:

Total Number of Nodes: is the number of nodes, both intermediate and terminal, which were generated during the search.

Total Number of Iterations: is the number of forward and backward moves conducted during the search.

Number of times it was necessary to solve an assignment problem: whenever the lower bound calculated at any particular node by the reduction method was less than the current upper bound, it was necessary to solve a linear assignment problem in order to improve the value of this lower bound. Naturally, this strategy is applicable only to the algorithm which calculates the lower bounds by the reduction method.

Fathoming Efficiency: is the ratio between the number of times the search was not pursued, as a result of the lower bound test, to the total number of times the lower bound test was applied.

Comparison between QAP3 and QAP7

We recall that the only difference between QAP3 and QAP7 is the method of calculating the lower bound as shown in IV.2. From Table IV.1, we notice that the solution times when using QAP7 were always less than those obtained when QAP3 was used.

Our observation about QAP3 is that it can face severe difficulties when the problem size increases. For example, problem 4005 was rerun with a starting upper bound of 289 and 100% fathoming in hope that this will speed up the search and there will be no need to restart the search. However, we had to terminate the problem after 50,400 iterations as we noticed that 11,539 nodes were generated but the number of nodes at various levels were 0 4 41 358 2411 4101 3715 1000 184 8 3 3, i. e., a substantial part of the tree was still to be searched and the estimated time for the completion of the search was about 15 minutes on the IBM 370/165. Also, the experience with 4006 was not anymore encouraging. It may be worth mentioning, however, that to the best of our knowledge, no exact solution to problem 4005 was known before.

Note that both QAP3 and QAP7 found the optimal solution of problems 4001 through 4004 and verified optimality. QAP3 and QAP7 found the optimal of problem 4005 but only QAP7 verified optimality (this is why QAP7 used more time than QAP3 for problem 4005). It is interesting to note that the concept of stepped fathoming was essential in the procedure because otherwise the computational effort would increase significantly.

TABLE IV.1 Summary of the Computational Experience with QAP3 and QAP7

Problem Number	Size	Algorithm	Total Number of Nodes	Total Number of Iterations	No. of times the LB was calculated	No. of times it was necessary to solve an AP	Fathoming Efficiency %	Value of Best Solution Obtained	Solutions Time (seconds)*
4001	5x5	QAP3	15	38	43	--	46.12	25	0.26
		QAP7	9	20	20	14	45.00	25	0.15
4002	6x6	QAP3	27	99	129	--	54.26	43	1.01
		QAP7	18	56	67	36	52.24	43	0.63
4003	7x7	QAP3	55	177	235	--	51.06	74	3.9
		QAP7	22	62	73	40	58.90	74	2.7
4004	8x8	QAP3	231	739	1005	--	50.25	107	15.3
		QAP7	52	179	235	141	52.77	107	9.6
4005	12x12	QAP3	6877	29282**	--	--	--	289	221.8
		QAP7	5724	24496	37531	26368	50.02	289	490.4
4006	15x15	QAP3	10071	40502***	--	--	--	618	--

*On an IBM 370/165

**No restarting was allowed

***The search was forced to termination

TABLE IV.2 Number of Nodes Generated at the Tree Levels

PROB. NUMB.	ALGORITHM	NUMBER OF ITERATIONS	TOTAL NUMBER OF NODES	LEVELS											
				1	2	3	4	5	6	7	8	9	10	11	12
4001	QAP3	38	15	0	4	5	3	3							
	QAP7	20	9	0	2	3	2	2							
4002	QAP3	99	27	0	6	13	4	2	2						
	QAP7	56	18	0	6	2	4	3	3						
4003	QAP3	177	55	0	7	21	21	3	1	2					
	QAP7	62	22	0	6	8	2	2	2	2					
4004	QAP3	739	231	0	8	40	109	58	10	3	3				
	QAP7	179	52	0	8	17	10	8	3	3	3				
4005	QAP3	29282*	6877	0	12	125	866	3367	1774	645	73	6	3	3	3
	QAP7	24496	5724	0	12	132	884	2539	1492	585	53	18	3	3	3

*Decision tree has not been exhausted

2) The Heuristic Procedure

Two versions of the Heuristic Procedure were tested.

a) QAH1: a heuristic based on the ideas mentioned in IV.4, except that it terminates once all elements have been located and the test of single exchanges is performed on the last located element only.

b) QAH2: Multi double exchanges are performed after single exchanges cease to provide further improvements.

Computational experience with the above two heuristic procedures and their comparison with some well known heuristic methods is summarized in Table IV.3. The results of the procedures of (Hillier [15], Hillier and Connors [16], Craft [1,4], and biased sampling [21]) have been published in (Nugent, et. al., [21]), and together with the results of (Neghabat [20]) are reproduced here. QAH1 and QAH2 compare favorably with these methods, both in the quality of solution obtained and the computational time. The column designating the best solution known so far is based on either true optimal or on the best (nonreproducible) result of the biased sampling method* of (Nugent, et. al, [21]).

*QAH2 obtained a solution with objective 1299 which is less than the best result known so far of 1304 for problem 40?7.

TABLE IV.3 Experimental Comparisons With Some Existing Heuristic Algorithms

PROBLEM NUMBER	PROB-LEM SIZE	HILLIER		HILLIER-CONNORS		CRAFT		BIASED SAMPLING			NEGHABAT		QAH1		QAH2		BEST KNOWN COST
		AVE. COST	AVE* TIME	AVE COST	AVE* TIME	AVE COST	AVE* TIME	AVE. COST (minimum)	AVE. COST (mean)	AVE.* TIME	COST	TIME*	COST	TIME **	COST	TIME **	
4001	5	27.6	0.5	29.4	0.83	28.2	0.08	26.8	27.6	0.92	25	0.05	26	0.14	26	0.12	25***
4002	6	44.2	0.58	44.2	0.75	44.2	0.16	43.6	44.6	1,75	43	0.1	43	0.14	43	0.21	43***
4003	7	78.8	1.25	78.4	1	79.6	0.4	74.8	77.2	4.75	74	0.3	78	0.11	75	0.39	74***
4004	8	114.4	1.2	110.2	1.2	113.4	0.83	107.0	111.6	9.08	118	0.74	114	0.17	114	0.42	107***
4005	12	317.4	4.6	310.2	1.6	296.2	5.8	293.0	304.7	54.83	334	1.5	295	0.73	295	1.94	289***
4006	15	632.6	6.5	600.2	3.3	606	13.3	580.2	603.0	182.7	618	2.5	645	0.69	614	6.07	575
4007	20	1400.4	14	134.5	6.25	1339	44	1313.0	1339.0	576.3	1385	3.6	1345	1.71	1299	19.49	1304
4008	30	3267.2	33	3206.8	23.7	3189.6	262	3124.8	3189.6	3520	3260	4.5	3170	15.16	3125	71.99	3093

*Actual or equivalent time on IBM 360/65 in seconds

**Time on Univac 1108 in seconds

***Optimal Solution

CONCLUSION

In this section we have been concerned with solving the quadratic assignment problem. Our conclusion is that this problem is extremely difficult to solve exactly for several reasons, including the combinatorial nature of the problem, and the fact that the lower bounds calculated during a branch and bound scheme are generally very weak, leading to a lengthy search.

It was our hope to tighten the lower bounds by a duality based scheme as shown in IV.3. Even though this goal has not been achieved, we believe that further work along these lines for raising the lower bound is essential for obtaining exact and "qualified" suboptimal solutions via branch and bound.

Based on our experience, it is feasible to solve quadratic assignment problems with sizes up to 12 x 12 exactly by the proposed exact procedure. However, for larger problems, it is possible to obtain good results in reasonable times by the use of the proposed heuristic (QAH2) or its curtailed version (QAH1).

REFERENCES

1. Armour, G. C. and E. S. Buffa, "A Heuristic and Simulative Approach to Relative Location of Facilities," Management Science, Vol. 9, pp. 294-309, 1963.

2. Bazaraa, M. S. and J. J. Goode, "A Survey of Various Tactics for Generating Lagrangian Multipliers in the Context of Lagrangian Duality," Georgia Institute of Technology, August 1974.

3. Breuer, M. A., "The Formulation of Some Allocation and Connection Problems as Integer Problems," Naval Research Logistics Quarterly, Vol. 13, pp. 83-95, 1966.

4. Buffa, E. S., G. C. Armour, and T. E. Vollman, "Allocating Facilities with CRAFT," Harvard Business Review, Vol. 42, pp. 136-158, 1964.

5. Dorris, A. L., "The Utility of Optimization Techniques in the Design of Man-Machine Systems," Masters Thesis, Georgia Institute of Technology, 1971.

6. Francis, R. L. and J. A. White, "Facility Layout and Location: An Analytical Approach," Prentice-Hall, 1974.

7. Gaschutz, G. K. and J. H. Ahrens, "Suboptimal Algorithm for the Quadratic Assignment Problem," Naval Research Logistics Quarterly, Vol. 15, pp. 49-62, 1968.

8. Gavett, J. W. and N. V. Plyter, "The Optimal Assignment of Facilities to Locations by Branch and Bound," Operations Research, Vol. 14, pp. 210-232, 1966.

9. Gilmore, P. C., "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem," SIAM Journal on Applied Mathematics, Vol. 10, pp. 305-313, 1962.

10. Graves, G. W. and A. B. Whinston, "An Algorithm for the Quadratic Assignment Problem," Management Science, Vol. 17, pp. 453-471, 1970.

11. Hanan, M. and J. Kurtzberg, "A Review of the Placement and Quadratic Assignment Problems," SIAM Review, Vol. 14, pp. 324-342, 1972.

12. Heider, C. H., "An n-step, 2-variable Search Algorithm for the Component Placement," Naval Research Logistics Quarterly, Vol. 20, pp. 699-724, 1973.

13. Held, M. and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," Mathematical Programming, Vol. 1, pp. 6-25, 1971.

14. Held, M., P. Wolfe, and H. D. Crowder, "Validation of Subgradient Optimization," Mathematical Programming, Vol. 6, pp. 62-88, 1974.

15. Hillier, F. S., "Quantitative Tools for Plant Layout Analysis," The Journal of Industrial Engineering, Vol. 14, pp. 33-40, 1963.

16. Hillier, F. S. and M. M. Connors, "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities," Management Science, Vol. 13, pp. 42-57, 1966.

17. Koopmans, T. C. and M. Beckman, "Assignment Problems and the Location of Economic Activities," Econometrica, Vol. 25, pp. 53-76, 1957.

18. Land, A. H., "A Problem of Assignment with Interrelated Costs," Operational Research Quarterly, Vol. 14, pp. 185-198, 1963.

19. Lawler, E. L., "The Quadratic Assignment Problem," Management Science, Vol. 9, pp. 586-599, 1963.

20. Neghabat, F., "An Efficient Equipment Layout Algorithm," Operations Research, Vol. 22, pp. 622-628, 1974.

21. Nugent, C. E., T. E. Vollman, and J. Ruml, "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations," Operations Research, Vol. 16, pp. 150-173, 1968.

22. Pegels, C. C., "Plant Layout and Discrete Optimizing," International Journal of Production Research, Vol. 5, pp. 81-92, 1966.

23. Pierce, J. F. and W. B. Crowston, "Tree Search Algorithms for Quadratic Assignment Problems," Naval Research Logistics Quarterly, Vol. 18, pp. 1-36, 1971.

24. Steinberg, L., "The Backboard Wiring Problem: A Placement Algorithm," SIAM Applied Mathematics, Vol. 3, pp. 37-50, 1961.

25. Vollman, T. E., C. E. Nugent, and R. L. Zartler, "A Computerized Model for Office Layout," The Journal of Industrial Engineering, Vol. 19, pp. 321-329, 1968.

26. Whitehead, B. and M. Z. Elders, "An Approach to the Optimum Layout of Single Story Buildings," Architect's Journal, Vol. 139, pp. 1373-1380, 1964.

V. THE TRAVELING SALESMAN PROBLEM

As discussed in Section II, the traveling salesman problem is a special important case of the quadratic assignment problem. In this section, the dual of the problem is formulated and then solved by a subgradient optimization technique. In the presence of a duality gap, a branch and bound scheme is devised to find optimal and suboptimal tours.

V.1 Formulation of the Problem and its Dual

Suppose that a salesman is given a list of n cities, where each pair of cities, i and j, is connected by a link of length c_{ij} (if cities i and j are not directly linked, let $c_{ij} = +\infty$). The problem is then to find a tour of minimal length which the salesman can follow so as to visit each city exactly once, and then return to where he started.

Let x_{ij} be a variable associated with link ij. x_{ij} is equal to 1 if the salesman travels from city i to city j and is 0 otherwise. The problem can be stated as follows:

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} c_{ij}x_{ij}$$

$$\text{Subject to} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} = 1 \qquad i = 1, 2, 3, \ldots, n \qquad \text{V.1}$$

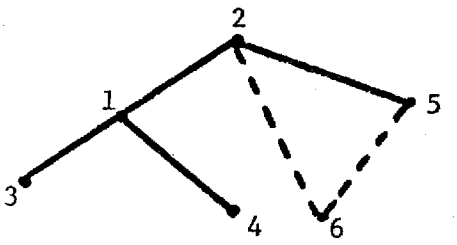
$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1 \qquad j = 1, 2, 3, \ldots, n \qquad \text{V.2}$$

x_{ij} is 0 or 1 $\qquad i = 1, \ldots, n;\ j = 1, \ldots, n \qquad$ V.3

No subtours

Since the restrictions V.1, V.2, and V.3, by themselves, may allow the meaningless answer of having two or more disjoint subtours, a restriction eliminating subtours must be explicitly introduced. There are various methods in the literature for enforcing this restriction. We will adopt the 1-tree method of (Held and Karp [7]).

Let us begin by reviewing the definition of a 1-tree [7]. A 1-tree of the graph composed of the n cities and their interconnecting links is a tree which spans n - 1 cities, together with two distinct links from the remaining city, to any two of the n - 1 cities spanned by the tree. Therefore, by definition, a 1-tree contains exactly one cycle. Of course, the number of links in the cycle is less than or equal to n. A 1-tree then corresponds to a tour if and only if the cycle formed by the 1-tree has n links. A 1-tree which is composed of a tree covering nodes 1, 2, 3, 4, and 5 plus two links from node 6 to the tree is shown below.



An Example of a 1-Tree

Now note that if constraints V.1, V.2, and V.3 are satisfied then there is at least one cycle. If there is only one cycle then it is a tour. By definition of the 1-tree, it admits exactly one cycle. Therefore, constraints V.1, V.2, and V.3, coupled with the constraint that the vector x forms a 1-tree, admit no subtours. Therefore, the traveling salesman problem can be formulated as follows:

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} c_{ij}x_{ij}$$

$$\text{Subject to:} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} = 1 \qquad i = 1, 2, \ldots, n$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

$$x \in X$$

where X is the set of 1-trees spanning the graph, i. e.,

$X = \{(x_{11}, \ldots, x_{1n}, \ldots, x_{nn}) : x_{ij} = 0 \text{ or } x_{ij} = 1 \ (i \neq j)$, and the links ij with $x_{ij} = 1$ form a 1-tree$\}$.

Note that there are very efficient procedures for finding a 1-tree with minimal cost [5,11]. This will involve pulling one node out of the n nodes, which we will refer to as the initial node, and then:

1. finding a tree of minimal length which spans the remaining n - 1 nodes
2. finding two links of minimal length which connect the initial node to the spanning tree.

Any node can be used as the initial node, and we will discuss later a procedure for making a "good" choice of the initial node. Calculating a 1-tree with minimal cost, in the presence of restrictions V.1, and V.2, namely the traveling salesman problem, is indeed a difficult problem. If we handle the restrictions V.1, and V.2 via lagrangian multipliers (dual variables) and put them into the objective function, we obtain the following dual formulation.

Dual Problem

Maximize $\Theta(u,v)$

u and v are unrestricted

where,

$$\Theta(u,v) = \underset{x \in X}{\text{Minimum}} \left\{ \sum_j \sum_i c_{ij}x_{ij} + \sum_i u_i\left(\sum_j x_{ij} - 1\right) + \sum_j v_j \left(\sum_i x_{ij} - 1\right) \right\}$$

$$= \underset{x \in X}{\text{Minimum}} \left\{ \sum_j \sum_i (c_{ij} + u_i + v_j)x_{ij} - \sum_i u_i - \sum_j v_j \right\}$$

$$= -\sum_i u_i - \sum_j v_j + \underset{x \in X}{\text{Minimum}} \sum_j \sum_i (c_{ij} + u_i + v_j)x_{ij}$$

Note that evaluating $\Theta(u,v)$, for fixed values of u and v is an easy task since it involves finding a minimal 1-tree, where the cost c_{ij} of the link ij is replaced by $c_{ij} + u_i + v_j$. Also note that in finding the minimal 1-tree, we are not concerned about the orientation of the links. In other words, we are actually searching for a minimal-tree, rather than an arborescence. This simplifies the problem of evaluating $\Theta(u,v)$ considerably. Of course, we may think of the vectors u and v as penalties for violating the proper orientation of the links. So if when calculating the minimal 1-tree, too many links "left" node i we will attempt to "penalize" this deficiency in the next iteration by increasing u_i. Similarly, if too many links "entered" node j, then we will attempt to "penalize" this deficiency by increasing v_j.

Simplification of the Symmetric Case

In the symmetric case, as long as the vector x forms a 1-tree, and since the orientation of links is not important, we no longer require that the number of links leaving node i is 1, and the number of links entering node i is 1, as long as the number of links incident to node i is equal to 2. The primal problem can thus be reformulated as follows:

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} c_{ij}x_{ij}$$

$$\text{Subject to} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} + \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ji} = 2 \qquad \text{for } i = 1, 2, \ldots, n$$

$$x \in X$$

where X is the collection of 1-trees. The above restrictions ensure forming a tour. If the tour is not "oriented," it can be made so, by simply reorienting some of the links, without affecting the cost. The dual of the above problem is given below, with only one set of dual variables given by the vector u.

Maximize $\quad \theta(u)$

u is unrestricted

where,

$$\theta(u) = \underset{x \in X}{\text{Minimum}} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{n} u_i \left(\sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} + \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ji} - 2 \right)$$

$$= -2 \sum_{i=1}^{n} u_i + \underset{x \in X}{\text{Minimum}} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} (c_{ij} + u_i + u_j) x_{ij}$$

From now on we will concentrate on the nonsymmetric case with dual variables u and v. For the symmetric case, the dual problem can be simplified as illustrated above, with no need for the dual vector v.

V.2 Maximization of θ

Let the collection of 1-trees with a fixed initial node be $x_1, x_2, \ldots, x_t$. Then $\theta(u,v)$ can be written as follows:

$$\theta(u,v) = \underset{x \in X}{\text{Minimum}} \sum_i \sum_j c_{ij}x_{ij} + \sum_i u_i \left(\sum_j x_{ij} - 1 \right) + \sum_j v_j \left(\sum_i x_{ij} - 1 \right)$$

$$= \underset{x \in X}{\text{Minimum}} \; cx + ug(x) + vh(x)$$

$$= \underset{1 \le j \le t}{\text{Minimum}} \; cx_j + ug(x_j) + vh(x_j)$$

where $g_i(x) = \sum_j x_{ij} - 1$ and $h_i(x) = \sum_j x_{ji} - 1$ for $i = 1, 2, \ldots, n$. Note that θ is the minimum of a finite number of affine functions and is hence a piecewise linear concave function which is finite everywhere.

Several procedures for maximizing θ are available. Here a subgradient optimization procedure similar to that of (Held and Karp [8]) is used. Given a vector (u,v), a minimal 1-tree x is found, where the cost of link ij is $c_{ij} + u_i + v_j$. (u,v) is updated by moving a discrete step along the subgradient $\{g(x), h(x)\}$ and the process is repeated. Motivation, convergence, and geometrical aspects of the procedure can be found in [8], [9], and [1].

The initial node used throughout the algorithm is determined as follows. n minimal 1-tree problems with different initial nodes are solved with $u = v = 0$. The initial node whose minimal 1-tree has maximal objective is used. The following is a precise statement of the ascent algorithm used for maximizing the dual function (with obvious modification for the symmetric problem).

Subgradient Optimization of the Dual

Initialization Step

Choose $u_1 = v_1 = 0$. Choose a sequence of stepsizes $(\lambda_1, \lambda_2, \ldots, \lambda_t)$ where $\lambda_{j+1} < \lambda_j$ for $j = 1, 2, \ldots, t-1$. Choose the maximum number of iterations ν_{max} allowed without improving the incumbent θ^*. Let $j = k = 1$, $\nu = 0$, $u^* = u_1$, $v^* = v_1$, and $\theta^* = -\infty$. Go to Step 1.

Step 1 Finding $\theta(u_k, v_k)$

Solve the following minimal 1-tree subproblem.

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{i=1}^{n} (c_{ij} + u_{ik} + v_{jk}) x_{ij}$$

Subject to: $x \in X$

where $u_k = (u_{1k}, \ldots, u_{nk})$ and $v_k = (v_{1k}, \ldots, v_{nk})$.

Let x_k be an optimal 1-tree. If x_k is a directed tour, then stop, the optimal primal and dual solutions have been found. Otherwise, let $\Theta(u_k,v_k) = \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} (c_{ij} + u_{ik} + v_{jk})x_{ij} - \sum_{i=1}^{n} u_{ik} - \sum_{j=1}^{n} v_{jk}$.

If $\Theta(u_k,v_k) > \Theta^*$, then let $\nu = 0$, replace Θ^* by $\Theta(u_k,v_k)$, replace (u^*,v^*) by (u_k,v_k), x^* by x_k, and go to Step 2. If on the other hand, $\Theta(u_k,v_k) \leq \Theta^*$, replace ν by $\nu+1$. If $\nu = \nu_{max}$, go to Step 3. If $\nu < \nu_{max}$, go to Step 2.

Step 2 (Update dual variables)

Let $u_{k+1} = u_k + \lambda_j g(x_k)$ and $v_{k+1} = v_k + \lambda_j h(x_k)$. Replace k by k+1 and repeat Step 1.

Step 3 (reduce stepsize)

If $j = t$ then stop with Θ^* and (u^*,v^*). Otherwise replace j by j+1 and ν by 0. Let $u_{k+1} = u^* + \lambda_j g(x^*)$ and $v_{k+1} = v^* + \lambda_j h(x^*)$. Replace k by k+1 and repeat Step 1.

V.3 The Branch and Bound Scheme

So far we have described a procedure for finding the optimal (or near optimal) dual solution. In the process, an optimal tour for the traveling salesman problem may be found. If this were not the case, a branch and bound scheme for finding an optimal directed tour will be described. It is important to note that even in this case, optimizing the dual function played the following two significant roles:

1. Provided a "tight" lower bound on the optimal traveling salesman problem and hence making the notion of "qualified" suboptimal solutions attractive.
2. Rather than using the cost c_{ij} of link ij in the branch and bound scheme, c_{ij} is replaced by $c_{ij} + u_i + v_j$, where u and v are the

optimal dual variables, thus resulting in a "well-conditioned" cost matrix. The computational effort for finding the optimal tour, using the new cost matrix $c_{ij} + u_i + v_j$ is considerably smaller than the effort to be expended, had the original cost matrix been used in a branch and bound scheme. This is due to the fact that the new cost matrix $(c_{ij} + u_i + v_j)$ is "conditioned" in the sense that an optimal 1-tree with the above matrix is "close" to the optimal traveling salesman tour.

Consider now the following traveling salesman problem:

$$\text{Minimize} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{i=1}^{n} (c_{ij} + u_i + v_j)x_{ij}$$

$$\text{Subject to:} \quad \sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} = 1 \qquad \text{for } i = 1, 2, \ldots, n$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1 \qquad \text{for } j = 1, 2, \ldots, n$$

No Subtours

Note that an optimal tour obtained by the above problem is also an optimal tour of the original traveling salesman problem.

General Framework

A list of distinct nodes $i_1, i_2, i_3, \ldots, i_k$ and corresponding links $(i_1,i_2), \ldots, (i_{k-1},i_k)$ form a partial solution. Since the nodes are distinct, no subtours are a part of the partial solution. If $k = n$, then a complete tour is obtained by adding the link (i_k,i_1). Suppose that we are given a partial solution, and attempt to find a feasible completion. A lower bound on the cost of completing this partial solution is computed, say B. If $B \geq C^*$, where C^* is the cost of the best tour known so far, then the partial solution is fathomed. If $B < C^*$ then an additional link (i_k,i_{k+1}) is added to the partial tour where i_{k+1} is not equal to i_j for $j = 1, 2, \ldots, k$.

Calculation of the Lower Bound

Suppose that $i_1, i_2, \ldots, i_k$ form a partial solution. If (i_j, i_{j+1}) for $j = 1, 2, \ldots, k-1$ are to be a part of a tour, then the remaining nodes must themselves form a "string", which is connected to nodes i_1 and i_k. Since the "string" is a tree spanning the remaining nodes, then its cost must be greater or equal to the minimal cost of all trees spanning these nodes. Also, the cost of connecting the partial solution to this open loop is greater or equal to the sum of the following:

1. Minimal cost of (available) links leaving node i_k
2. Minimal cost of (available) links entering node i_1

We use the word available, since links (i_j, i_1) for $j = 2, \ldots, k$ and (i_k, i_j) for $j = 1, 2, \ldots, k-1$ are not available, because their use will form a subtour. Also, as the search continues, some other links will be prohibited from usage. From this it is clear that a lower bound on the cost of the partial solution, plus the cost of its completion, is given by:

$B = C_1 + C_2 + C_3$, where

C_1 : the actual cost of the partial solution

$$C_1 = \sum_{j=1}^{k-1} (c_{i_j i_{j+1}} + u_{i_j} + v_{i_{j+1}})$$

C_2 : the cost of the minimal tree spanning the remaining nodes, where the cost of link ij is $c_{ij} + u_i + v_j$.

C_3 : Minimal cost of "connecting" the partial solution to the spanning tree.

$$C_3 = \underset{j \notin A}{\text{Minimum}} (c_{i_k j} + u_{i_k} + v_j) + \underset{j \notin B}{\text{Minimum}} (c_{j i_1} + u_j + v_{i_1})$$

where A is the set of nodes which i_k is not eligible to enter, and B is the set of nodes which are not eligible to enter node i_1.

An ascent scheme could have been used to calculate the lower bound as in [8]. This is not done here, however, since the lower bound obtained above is easy to obtain and seems to be tight enough.

Continuation of the Search

1. Fathoming (Backward Move)

Suppose that we have a partial tour $i_1, i_2, \ldots, i_k$; i. e., $x_{i_1 i_2} = x_{i_2 i_3} = \ldots, = x_{i_{k-1} i_k} = 1$. The level of the search is called $k - 1$. The lower bound B on the cost of the partial solution plus its completion is calculated as shown above. If $B \geq C^*$, where C^* is the current best known cost of a complete tour, then the partial solution is fathomed. The last move from i_{k-1} to i_k is prohibited, i. e., $x_{i_{k-1} i_k} = 0$. A new bound B' on completing the cost of the partial tour $i_1, \ldots, i_{k-1}$ with the additional restriction that $x_{i_{k-1} i_k} = 0$ is computed. This is done in the same manner as before, except of course that i_k now belongs to the remaining nodes and the link $i_{k-1} i_k$ cannot be used. If $B' \geq C^*$, then the partial solution $i_1, i_2, \ldots, i_{k-1}$, with the added restriction $x_{i_{k-1} i_k} = 0$ can lead to no improved solutions. This means that all possibilities at level $k - 1$ are exhausted. This case will be called a strong fathoming, and the level of the tree is reduced by 1 unit. In this case $x_{i_{k-1} i_k} = 0$ is not prohibited any more, whereas $x_{i_{k-2} i_{k-1}}$ is forced to be zero. The process is then repeated. If on the other hand $B' < C^*$, which will be called weak fathoming, then we seek to find a node j such that $x_{i_{k-1} j} = 1$. This step will be discussed in more detail in the forward move below.

2. Forward Move

If the lower bound is less than C*, then it may be possible to find a completion of the partial solution with an objective less than C*. Suppose that the current level of the tree is k - 1, where at level k - 1 we have $x_{i_{k-1}i_k} = 1$. Now we seek a node $i_{k+1} \neq i_j$ for j = 1, 2, ..., k-1 and the move $i_k i_{k+1}$ is not prohibited from previous fathoming (if such a node i_{k+1} is not found then the partial solution is fathomed and the level of the tree is reduced). This rule is used: choose i_{k+1} which minimizes $c_{i_k j} + u_{i_k} + v_j$ for all available j. The level of the tree is increased by 1, a lower bound is calculated, and the process is repeated. Needless to say, if the level is n - 1, then i_n is joined with i_1 and the cost of the link $i_n i_1$ is added to the cost of the partial tour. If the total cost is less than C*, then C* is replaced by the total cost, and the tour is stored as the best tour obtained so far.

3. Termination

We have described forward and backward progress of the search. If the level of the search ever reaches value zero, then we stop. This means that all possible completions of $x_{i_1 i_2} = 1$ and $x_{i_1 i_2} = 0$ have been enumerated, i. e., all the search space has been exhausted. The stored tour and C* give the optimal solution.

Summary of the Algorithm

Initialization Step

Let P(i) = $\emptyset$ for all i = 1, 2, ..., n and let C* = ∞. Choose node i_1 (according to the procedure described earlier) as the initial node in the maximization of θ. Let k = 0 and go to Step 1.

Step 1 (Forward Move)

If $k = n - 1$, replace C_1 by $C_1 + (c_{i_n i_1} + u_{i_n} + v_{i_1})$. If $C_1 < C^*$, then replace C^* by C_1, store the complete solution $i_1, \ldots, i_n, i_1$ and go to Step 2. If $C_1 \geq C^*$, go to Step 2. If $k < n - 1$, calculate a lower bound B on all completions of the partial solution at hand, where $B = C_1 + C_2 + C_3$, C_1, C_2, and C_3 are calculated as described above, $A = P(i_k) \cup \{i_1, \ldots, i_{k-1}, i_k\}$ and $B = \{i_1, i_2, \ldots, i_k\}$. If $B \geq C^*$, go to Step 2. Otherwise, pick $i_{k+1} \notin A$ such that $c_{i_k i_{k+1}} + u_{i_k} + v_{i_{k+1}} =$ Minimum $(c_{i_k j} + u_{i_k} + v_j)$. Replace k by k + 1 and repeat Step 1.
$j \notin A$

Step 2 (Fathom)

Replace $P(i_{k-1})$ by $P(i_{k-1}) \cup \{i_k\}$. Calculate a lower bound B on all completions of the partial solution $i_1, i_2, \ldots, i_{k-2}, i_{k-1}$. If $B \geq C^*$, go to Step 3. Otherwise, choose a node $j \notin A$ such that $c_{i_{k-1} j} + u_{i_{k-1}} + v_j = \text{Minimum } \{c_{i_k i} + u_{i_{k-1}} + v_i : i \notin A\}$. This forms a new partial solution, where i_k is replaced by j. Go to Step 1.

Step 3 (Strong Fathoming)

i_{k-1} is placed in $P(i_{k-2})$. $P(i_{k-1})$ is replaced by $\emptyset$, and k is replaced by k - 1. If k = 0, go to Step 4, otherwise go to Step 2.

Step 4 (Termination)

Search has been completed. Optimal cost is C^* and its corresponding tour is $i_1, i_2, \ldots, i_n, i_1$. Stop.

V.4 Stepped Fathoming

Rather than attempting to fathom a partial solution when the lower bound B on its completion is greater than or equal to the best known C^*, we may fathom if B is "close" to C^*. This will enable us to fathom quickly

and hence reduce the portion of the search tree explicitly enumerated. This procedure can be controlled either to obtain suboptimal solutions, with any desired degree of accuracy or to obtain optimal solutions. In both cases the computational effort will be significantly reduced. The following two methods are proposed for implementing this notion.

Method 1

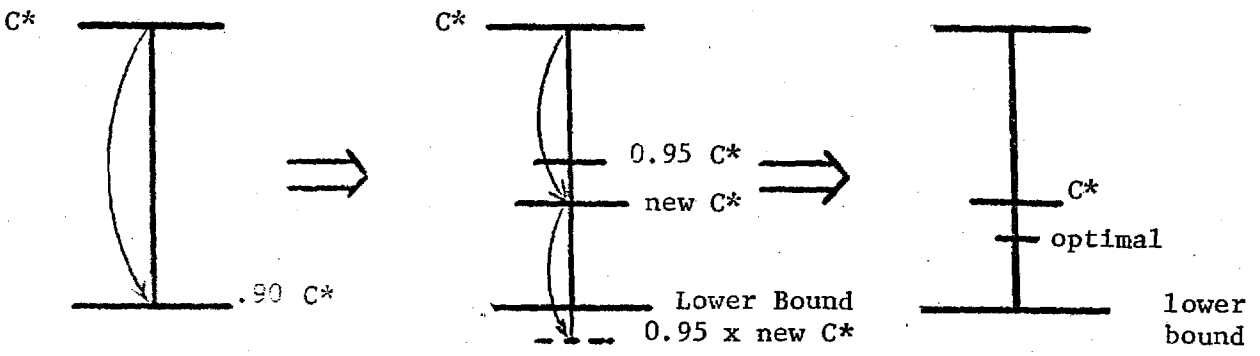
Suppose that a partial solution is fathomed if $B \geq \alpha C^*$, where $\alpha \in (0,1]$. In this case, the partial solution is abandoned if there is no hope that it leads to a cost which is better than αC^*. The objective of this simple strategy is clear. We want to fathom the partial solution quickly, even if it may lead to a slight improvement. As the reported computational experience shows, this simple strategy speeds the computational effort considerably. Of course, as a new C^* is found, then we fathom whenever the bound is greater or equal to α times the new C^*. The procedure continues until we cannot find a feasible solution with an objective less than αC^*. So we have a feasible tour with objective C^*, coupled with the statement that the optimal objective is greater than or equal to αC^*. Needless to say, if αC^* is less than the overall lower bound, which is Max $\theta(u,v)$, then there is no need to complete the search, since we know that there are no solutions with objective less than αC^*.

Choice of α

If α is small, fathoming will speed up considerably resulting in a small compuational effort. But on the other hand, the quality of the best obtained feasible solution is not guaranteed. A tradeoff between the desired solution quality and the computational effort expended exists. We recommend values of $\alpha \geq 0.93$ depending on the accuracy required.

Exact Solution by Stepped Fathoming

A slight modification of the above scheme can be used to find the exact optimal solution. Suppose that an initial α_1 is used, say $\alpha_1 = 0.90$. Eventually we achieve a final C* and cannot find a feasible solution with objective less than 0.90 C*, i. e., 0.9 C* is a lower bound on all solutions. We either stop here, or else switch to $\alpha_2 \geq \alpha_1$, say $\alpha_2 = 0.95$. We start the search from the stored best solution corresponding to C*, which also includes all the prohibited visitations. We are either able to find a feasible solution with objective less than 0.95 C* (none less than 0.9 C* exist), and repeat the process, or else conclude that none exist. We may then switch to 97% fathoming and eventually 100% fathoming. This process is illustrated below and proved very efficient from a computational point of view.



no solution with objective < 0.9 C* exists

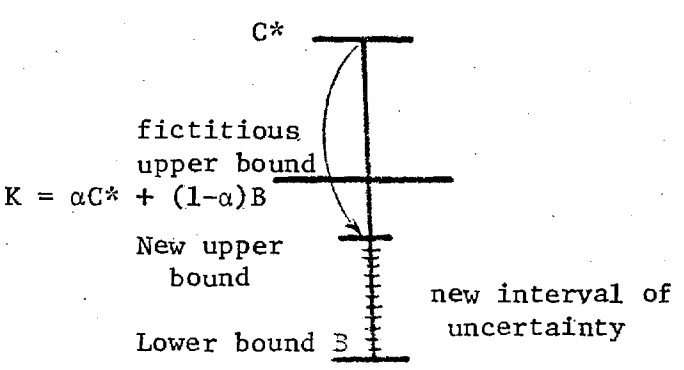
A solution with objective < 0.95 C* found

90% fathoming 95% fathoming 100% fathoming

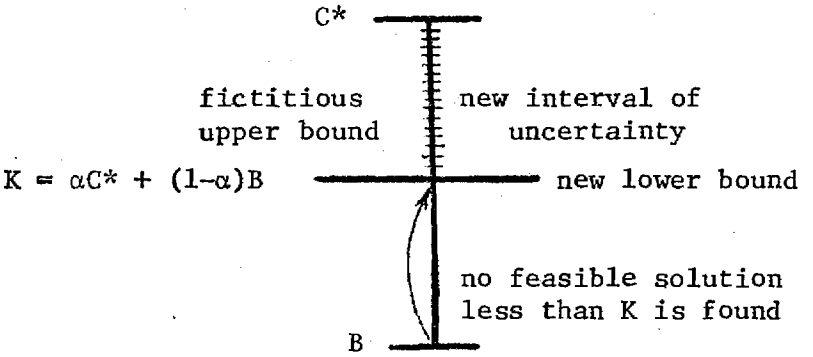
Method 2

At each stage of the algorithm, we have an upper bound C*. A lower bound on the overall problem, namely Max $\theta(u,v)$ is also known. Suppose we fathom on $K = \alpha C^* + (1 - \alpha)B$, where $\alpha \in (0,1]$. Two cases are possible. In the first case, we will be able to find a tour with objective less than K. The objective of this new solution becomes the new upper bound

C* and the process is repeated. In the second case, we will not be able to find such a solution. This automatically implies that there are no solutions with objective less than K, and hence K itself is the new lower bound. The process is then repeated. The gap between the lower and upper bounds is continuously reduced either by lowering the upper bound or by raising the lower bound. When the difference between them is smaller than a prescribed tolerance, we either stop, or try to find solutions with objective less than C* itself, i. e., we switch to $\alpha = 1$. In the former case we obtain a "qualified" suboptimal solution, and in the latter case we obtain the optimal solution.



Upper bound reduced



Lower bound reduced

Choice of α

α is any number in the interval (0,1]. Of course, if α is close to 1, then we are in effect fathoming on a number very close to C*, and the search will not speed considerably. On the other hand, if α is close to zero, then we only fathom if we obtain a feasible solution very close to the overall lower bound. In this case, fathoming will be fast, but it is likely not to obtain feasible solutions less than K. The value $\alpha = 0.5$ is recommended so that the distance of uncertainty is halfed at each time.

Computational effectiveness of stepped fathoming will be clear in the next section. For more details on the application of stepped fathoming in tree search algorithms the reader may refer to [2].

V.5 Computational Experience

This section summarizes the computational experience with the algorithm discussed in V.3 for both symmetric and nonsymmetric problems. We attempted to solve some standard problems in the literature as well as some problems generated at random. The following three standard symmetric problems were solved:

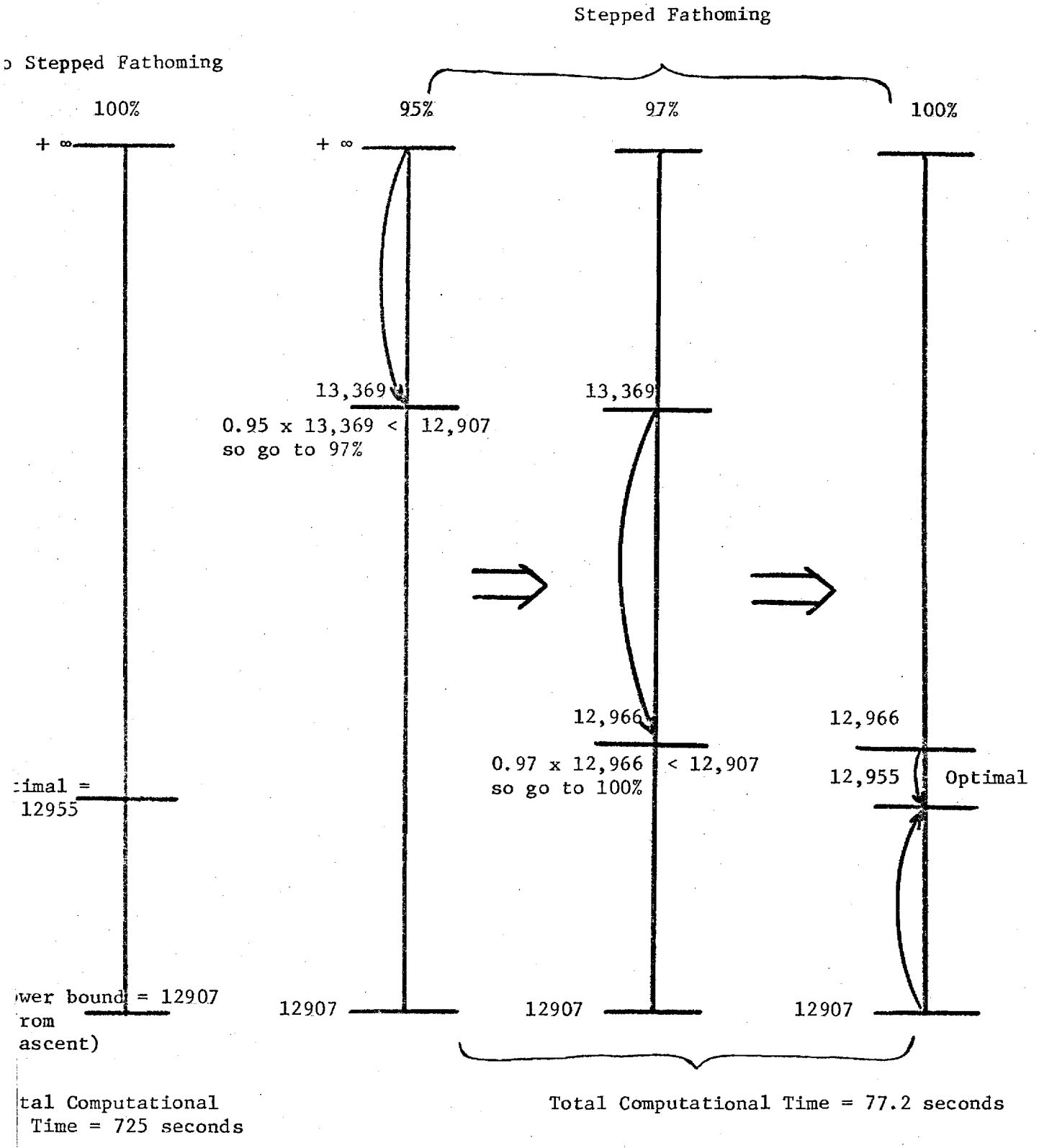
1. 25 city problem of (Held and Karp [6]).
2. 42 city problem of (Dantzig, Johnson, and Fulkerson [4]).
3. 57 city problem of (Karg and Thompson [10]).

In addition, several problems ranging from 30 to 80 cities have been generated at random from a uniform distribution with lower and upper limits of 100 and 3400. All computations were executed on a Univac 1108 machine and the reported times are in seconds.

During the course of dual optimization (or ascent) two sequences of stepsizes have been used, namely (5.0, 1.0, 0.2) and (5.0, 1.0). The stepsize is reduced if 20 trials (in some instances 30 trials were used) to improve the best known dual objective failed to do so. The sequence (5.0, 1.0, 0.2) with 30 failures managed to attain a higher dual value, of course, with more computational effort spent on the ascent. This usually resulted in less time expended in the branch and bound phase of the algorithm, however. From our experience it is very difficult to ascertain whether it is advantageous to have the smallest stepsize be 1.0, 0.2, or even less than 0.2. It is safe to say, however, that the extra effort used to reduce the stepsize to 0.2 is not lost as far as the overall computational effort is concerned.

In the branch and bound phase, the best multipliers of the ascent are used to obtain a modified cost matrix. As described in the last section, no ascent is used to calculate the bounds in this phase (as opposed to Held and Karp [8]). This, of course, increases the portion of the tree explicitly searched, but reduces the effort in calculating the bounds of completing the partial solutions. Also, the scheme of stepped fathoming was implemented and found very effective in reducing the portion of the tree to be enumerated.

We will use the 57 city problem to illustrate the concept of stepped fathoming. The results with and without employing the scheme are shown below. Note that the best dual objective of 12907 was obtained on ascent in 17 seconds. Using 100% fathoming (i. e., without stepped fathoming), the optimal tour with cost 12955 was obtained in an additional 708 seconds. The same optimal was obtained (and verified) using a stepped fathoming of 95%, 97%, and then 100% in an additional 60 seconds resulting in a total computational effort of 77 seconds.


Stepped Fathoming
Stepped Fathoming
100%
95%
97%
100%
+ ∞
+ ∞
13,369
13,369
0.95 x 13,369 < 12,907
so go to 97%
12,966
12,966
0.97 x 12,966 < 12,907
so go to 100%
12,955
Optimal
12907
12907
12907
Total Computational Time = 77.2 seconds

Results of the 25, 42, and 57 City Problems

Table V.1 summarizes the results of the 25, 42, and 57 city problems. The last column gives the computational time reported by (Held and Karp [8]) to obtain the optimal solutions (time in seconds on an IBM 360/91). Stepped fathoming with 95%, 97%, and 100% is used here to find and verify the optimal solutions. Note that in the 25 city problem no duality gap exists. A branch and bound phase was used, however, since an alternative optimal 1-tree which is not a tour was found on ascent.

The branch and bound procedure can be used to obtain suboptimal solutions with any prescribed apriori degree of accuracy. The computational effort will usually increase as a function of the accuracy desired. To illustrate, a 95% fathoming was used to solve the 42 city problem. A tour with cost 708 is obtained in 6 seconds. The percentage deviation of the optimal = 699 from the 95% solution 708 is 1.2% (less than the theoretical bound of 5%). Even if the true optimal solution were not known, then in 6 seconds we would have obtained a tour with cost 708 coupled with the statement that the optimal solution is greater than or equal to 694.

It is interesting to note, however, that $\alpha_1 > \alpha_2$ does not necessarily imply that quality of an α_1% solution will be better than an α_2% solution or that the computational effort of an α_1% solution will be larger than an α_2% solution. This is due to the fact that the quality of the solution and the computational effort depend on many other factors such as the lower bound from the ascent phase, the new cost matrix, the intermediate feasible solutions obtained, etc.

Symmetric Random Problems

The two methods of stepped fathoming discussed in V.4 were used to obtain suboptimal solutions of randomly generated problems of sizes ranging

TABLE V.1

Problem Size	Dual objective before ascent	Dual objective after ascent	Time for ascent (seconds)	Optimal primal objective	Time for branch and bound (seconds)	Total Computational time (seconds)	Computational time reported by Held and Karp
25	1453	1711	1.4	1711	0.2	1.6	12
42	629	694	5.3	699	22.5	27.8	54
57	11520	12907	17.2	12955	60.0	77.2	780

from 40 to 80 cities. In many cases, optimality is verified during the course of optimization since the primal solution equals the lower bound obtained from ascent. These are denoted by *.

Method 1

Stepped fathoming with 95%, 97%, and then 99% was used on problems of size 40, 50, and 60. On problems of size 70 a stepped fathoming of 95% and then 98% is used. Finally, a 95% fathoming is used in case of the 80 cities. We chose not to obtain 100% solutions because the computational effort increases significantly with very little improvement (or none at all) in the quality of solutions. The results of method 1 are summarized in Table V.2.

Method 2

The same random problems shown above were solved by method 2 where the interval of uncertainty is halfed at each stage until the difference between the lower bound and the upper bound (best known feasible solution) is less than or equal to 100. The quality of solutions and the computational effort is comparable to that of method 1. A summary of the results is shown in Table V.3.

Nonsymmetric Random Problems

For the nonsymmetric problem two sets of dual variables u_i's and v_i's must be used. For this reason the minimal 1-tree calculation usually consumes more effort. Since this calculation is repeated continuously during the ascent scheme, we may expect that the overall computational effort will be larger than the symmetric case. Also the increased complexity of the dual space makes it more difficult for the ascent scheme to reach the true maximum of the dual function.

TABLE V.2

Problem Size	Dual objective before ascent	Dual objective after ascent	Time for ascent (seconds)	Primal Solution	%age theoretical deviation from lower bound	%age actual deviation from lower bound	Time for branch and bound (seconds)	Total computational time (seconds)
40	7956	10573	8.9	10573*	1.0%	0.0%	0.8	9.7
40	7998	10615	7.8	10615*	1.0%	0.0%	1.1	8.9
50	7721	9462	18.0	9533	1.0%	0.8%	7.5	25.5
50	7819	9495	16.0	9496	1.0%	0.0%	1.1	17.1
60	8765	10643	20.0	10643*	1.0%	0.0%	1.7	21.7
60	8779	10528	23.1	10528*	1.0%	0.0%	1.6	24.7
70	11089	13287	35.0	13368	2.0%	0.6%	12.4	47.4
70	11255	13418	28.2	13603	2.0%	1.2%	15.1	43.3
80	11764	14365	41.8	14671	5.0%	2.1%	4.1	45.9
80	11707	14199	55.2	14521	5.0%	2.2%	7.0	62.2

TABLE V.3

Problem Size	Dual objective before ascent	Dual objective after ascent	Time for ascent (seconds)	Primal Solution	%age actual deviation from lower bound	Time for branch and bound (seconds)	Total computational time (seconds)
40	7956	10573	8.9	10573*	0.0%	1.5	10.4
40	7998	10615	7.8	10615*	0.0%	0.9	8.7
50	7721	9462	18.0	9533	0.8%	15.6	33.6
50	7819	9495	16.0	9496	0.0%	1.1	17.1
60	8765	10643	20.0	10643*	0.0%	0.5	20.5
60	8779	10528	23.1	10528*	0.0%	1.5	24.6
70	11089	13287	35.0	13368	0.6%	17.0	52.0
70	11255	13418	28.2	13501	0.6%	77.9	106.1
80	11764	14365	41.8	14436	0.5%	4.0	45.8
80	11707	14199	55.2	14678**	3.2%	8.0	63.2

**In this particular run the algorithm was not able to reduce the gap between the lower and upper bounds to 100 in 5 minutes, even though at least 96.8% quality solution is obtained after approximately 1 minute.

Computational results with random problems with n varying from 30 to 60 are demonstrated in the following two tables. For larger problems, both methods 1 and 2 were not able to reduce the gap between the lower and upper bounds to within 4 percent of the value of the upper bound in 5 minutes.

Method 1

Stepped fathoming 92%, 96%, and then 99% is employed for the first three problems whereas stepped fathoming 92% then 96% is used for the remainder of the problems. The results are shown in Table V.4.

Method 2

The same problems summarized above were run using method 2 of stepped fathoming where the interval of uncertainty is halfed at each iteration. The process is terminated when the difference between the lower and upper bounds is less than ε. The results are summarized in Table V.5.

From Tables V.4 and V.5 we see that methods 1 and 2 are comparable both from the quality of solution and computational time.

It is worthwhile noting that the procedure was not able to solve nonsymmetric problems with size 70 and 80 cities within 5 minutes, whereas other procedures in the literature are able to do so (see for example Bellmore and Malone [3]). Also, good quality solutions for nonsymmetric problems seem harder to get than those for symmetric problems.

Table V.4

Problem Size	Dual objective before ascent	Dual objective after ascent	Time for ascent (seconds)	Primal Solution	%age theoretical deviation from lower bound	%age actual deviation from lower bound	Time for branch and bound (seconds)	Total computational time (seconds)
30	4899	7427	10.5	7519	1%	1 %	26.8	37.2
30	4899	7427	10.4	7519	1%	1 %	26.7	37.1
40	6212	9383	19.2	9606*	1%	2.3%	11.4	30.6
40	6212	9383	19.9	9724	4%	3.4%	1.2	21.1
50	7448	9881	28.1	10059	4%	1.8%	12.5	
50	7446	9879	24.2	10057	4%	1.8%	14.1	38.3
60	8134	10871	31.2	11309	4%	3.9%	3.9	35.1
60	8132	10869	29.8	11308	4%	3.9%	3.3	33.1

*In this particular run the algorithm was not able to finish the 99% fathoming in 2 minutes.

TABLE V.5

Problem Size	Dual objective before ascent	Dual objective after ascent	Time for ascent (seconds)	Primal Solution	Termination criterion ε	%age actual deviation from lower bound	Time for branch and bound (seconds)	Total computational time (seconds)
30	4899	7427	12.4	7519	100	1.2%	13.7	26.1
30	4899	7427	11.3	7519	100	1.2%	13.8	25.1
40	6212	9383	19.6	9520	300	1.4%	38.5	58.1
40	6212	9383	20.0	9520	300	1.4%	39.2	59.2
50	7448	9881	26.2	10059	300	1.8%	49.6	75.8
50	7446	9879	26.8	10057	300	1.8%	51.5	78.3
60	8134	10871	31.2	11309	440	3.9%	1.9	33.1
60	8132	10869	28.3	11308	440	3.9%	1.6	29.9

CONCLUSION

We have described a duality based procedure for solving the traveling salesman problem. The computational experience with symmetric problems seems quite promising. However, for nonsymmetric problems there seems to be room for further improvement.

Subgradient optimization was used in this study as the main procedure for maximizing the dual. The performance of other procedures such as ascent, steepest ascent, decomposition, and search methods, remains to be tested.

REFERENCES

1. Bazaraa, M. S. and J. J. Goode, "A Survey of Various Tactics for Generating Lagrangian Multipliers in the Context of Lagrangian Duality," Georgia Institute of Technology, Atlanta, Georgia, August 1974.

2. Bazaraa, M. S. and A. N. Elshafei, "The Concept of Stepped Fathoming in Tree Search Algorithms," Georgia Institute of Technology, Atlanta, Georgia, June 1974.

3. Bellmore, M. and J. C. Malone, "Pathology of Traveling-Salesman Subtour-Elimination Algorithms," Operations Research, Vol. 19, pp. 278-307, 1971.

4. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson, "Solution of a Large Scale Traveling Salesman Problem," Operations Research, Vol. 2, pp. 393-410, 1954.

5. Dijkstra, E. W., "A Note on Two Problems in Connection with Graphs," Numerische Mathematik, Vol. 1, pp. 269-271, 1959.

6. Held, M. and R. M. Karp, "A Dynamic Programming Approach to Sequencing Problems," SIAM Journal on Applied Mathematics, Vol. 10, pp. 196-210, 1962.

7. Held, M. and R. M. Karp, "The Traveling Salesman Problem and Minimal Spanning Trees," Operations Research, Vol. 18, pp. 1138-1162, 1970.

8. Held, M. and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," Mathematical Programming, Vol. 1, pp. 6-25, 1971.

9. Held, M., P. Wolfe, and H. D. Crowder, "Validation of Subgradient Optimization," Mathematical Programming, Vol. 6, pp. 62-88, 1974.

10. Karg, L. L. and G. L. Thompson, "A Heuristic Approach to Solving Traveling Salesman Problems," Management Science, Vol. 10, pp. 225-248, 1964.

11. Kruskal, J. B., "On the Shortest Spanning Subtree of the Graph and the Traveling Salesman Problem," Proceedings of American Mathematical Society, Vol. 1, pp. 48-50, 1956.

PERSONNEL

This section presents the personnel involved in this research and the contributions they have made. It also gives a list of papers related to this research.

Principal Investigator

Dr. Mokhtar S. Bazaraa
Associate Professor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Contribution

Organization and coordination of the project. Development of the algorithms in Sections III, IV, and V. Review of the programming development.

Graduate Research Assistants

1. Virginia Ruth Kluge
   School of Industrial and Systems Engineering
   Georgia Institute of Technology

   #### Contribution

   Computer program of the initial solutions of the quadratic assignment and traveling salesman problems.

2. Peter H. Geddes
   School of Industrial and Systems Engineering
   Georgia Institute of Technology

   #### Contribution

   Applications of the quadratic set covering problem, quadratic assignment problem, and related problems.

3. Sinan S. Tumer
   School of Industrial and Systems Engineering
   Georgia Institute of Technology

   Contribution

   Programming of the quadratic set covering algorithm. Generation and running of problems.

Acknowledgement

Dr. Jamie J. Goode of the School of Mathematics, Georgia Institute of Technology, contributed to the development of the traveling salesman algorithms of Section V.

Dr. A. N. Elshafei of the Institute of Planning in Cairo, Egypt, contributed to the development of the quadratic assignment algorithms of Section IV.

List of Papers Related to Project

1. M. S. Bazaraa and A. N. Elshafei, "The Concept of Stepped Fathoming in Tree Search Algorithms," submitted for publication.

2. M. S. Bazaraa and A. N. Elshafei, "Heuristic and Exact Procedures of the Quadratic Assignment Problem," submitted for publication.

3. M. S. Bazaraa and J. J. Goode, "The Traveling Salesman Problem: A Duality Approach," submitted for publication.

4. M. S. Bazaraa, "Computerized Layout Design: A Branch and Bound Approach," submitted for publication.

APPENDIX

This appendix gives a listing of the programs developed for solving quadratic set covering problems, quadratic assignment problems, and traveling salesman problems. The programs are organized into 10 files. A brief description of each file and its elements is given below. This is followed by a complete listing of all the programs.

1. QSC

   A set of programs for solving quadratic set covering problems in the context of layout design by enumeration. The file consists of the following elements.

   a. QSC. ASSGN
   b. QSC. BBOUND
   c. QSC. COSTUN
   d. QSC. INASS
   e. QSC. INPUT
   f. QSC. MAIN
   g. QSC. NEWBOU
   h. QSC. OUTPUT
   i. QSC. TIME

2. BAB

   A set of programs for solving quadratic set covering problems by branch and bound. The file consists of the following elements.

   a. BAB. BBOUND
   b. BAB. COST

3. QAP5

   A set of programs for solving quadratic assignment problems by branch and bound. The file consists of the following elements.

a. QAP5. MAIN
b. QAP5. UPDATE
c. QAP5. LWBND
d. QAP5. ARRNG
e. QAP5. TIMOUT
f. QAP5. DTAIN

4. QAP4

A set of programs for solving quadratic assignment problems by branch and bound. The file consists of the following elements.

a. QAP4. MAIN
b. QAP4. ARRNG
c. QAP4. UPDATE
d. QAP4. LWBND
e. QAP4. MAX
f. QAP4. DTAIN
g. QAP4. TIMOUT

5. QAP7

A set of programs for solving quadratic assignment problems by branch and bound. The file consists of the following elements.

a. QAP7. MAIN
b. QAP7. DTAIN
c. QAP7. GARY
d. QAP7. HADY
e. QAP7. LWBND2

6. QAIN

A set of programs for providing a heuristic solution to the quadratic assignment problem. The file consists of the following elements.

a. QAIN. MAIN
b. QAIN. DTAIN

7. NEW QAP

A set of programs for solving the dual of the quadratic assignment problem by subgradient optimization. The file consists of the following elements.

a. NEWQAP. MAIN
b. NEWQAP. UVW
c. NEWQAP. DTAIN
d. NEWQAP. KNAP
e. NEWQAP. HADY
f. NEWQAP. GARY
g. NEWQAP. X

8. QAP-DATA

This is a file consisting of the data for the quadratic assignment problems attempted. The file consists of the following elements.

a. QAP-DATA. PROBLEM 1
b. QAP-DATA. PROBLEM 2
c. QAP-DATA. PROBLEM 3
d. QAP-DATA. PROBLEM 4
e. QAP-DATA. PROBLEM 5
f. QAP-DATA. PROBLEM 6
g. QAP-DATA. PROBLEM 7
h. QAP-DATA. PROBLEM 8

9. FATHOM

A set of programs for solving symmetric traveling salesman problems with subgradient optimization and then branch and bound. The file consists of the following elements.

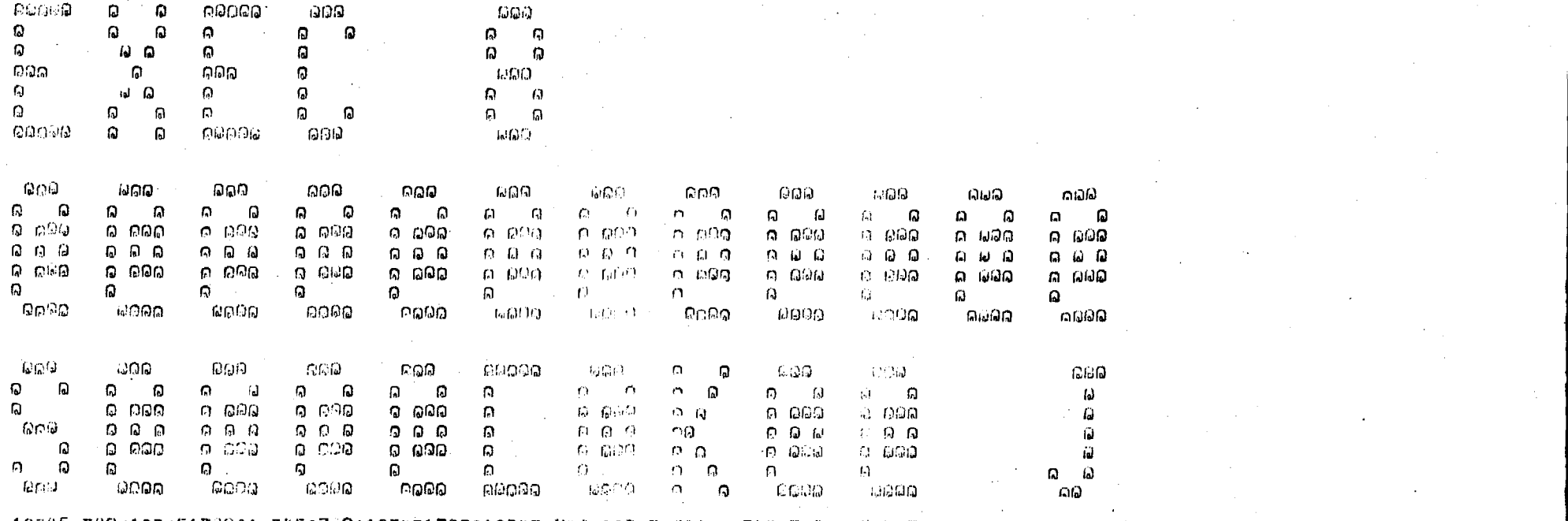
a. FATHOM. COPY
b. FATHOM. INPUT
c. FATHOM. INRAND
d. FATHOM. IO
e. FATHOM. MAIN

f. FATHOM. MINPT
g. FATHOM. MINPT 2
h. FATHOM. ORDER/NEW
i. FATHOM. ORDER/OLD
j. FATHOM. OUTPUT/BATCH
k. FATHOM. OUTPUT/DEMAND
l. FATHOM. SHOW/BATCH
m. FATHOM. SHOW/DEMAND
n. FATHOM. START
o. FATHOM. TIMOUT
p. FATHOM. TREE/NEW
q. FATHOM. TREE/OLD
r. FATHOM. VOGEL

10. NONSYM

A set of programs for solving nonsymmetric traveling salesman problems by subgradient optimization and then branch and bound. The file consists of the following elements.

a. NONSYM. CMINP3
b. NONSYM. INPUT
c. NONSYM. INRAND
d. NONSYM. IO
e. NONSYM. MAIN
f. NONSYM. MINPT
g. NONSYM. MINPT2
h. NONSYM. ORDER/OLD
i. NONSYM. OUTPUT/BATCH
j. NONSYM. OUTPUT/DEMAND
k. NONSYM. SHOW/BATCH
l. NONSYM. SHOW/DEMAND
m. NONSYM. START
n. NONSYM. TIMOUT
o. NONSYM. TREE



```
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890**SEQ NO. 00
CRDN 1ERON,011A0930,RARDIN,10,1000                                    CR1
CHECK CHG AND EXCESS LINES TO E-24-200
QELF,IDL MAIN
ELTOT/ RL1B70 02/21-16:58:42-(,0)
000001      000   QELF,SI TUNER-S*QSC.ASSGN,,,1011201101112
000002      000         SUBROUTINE ASSGN(KOD,COST)
000003      000   C
000004      000   C         ROUTINE TRIES TO ASSIGN UNASSIGNED OBJECTS
000005      000   C         KOD   :=1 IF COMPLETE ASSIGNMENT HAS BEEN MADE
000006      000   C               =-1 OTHERWISE
000007      000   C
000008      000   C
000009      000         INCLUDE BLANK
000010      000         INCLUDE INPOUT
000011      000   C         BEGIN TO ASSIGN ELEMENTS INTO THE LOCATIONS
000012      000   C         IELT=ELEMENT NUMBER
000013      000   C
000014      000         IELT=IAS+1
000015      000         NA=NAREA(IELT)
000016      000   122   IF(IELT.GT.1) IL=ILAST(IELT-1)
000017      000         IF(IELT.EQ.1) IL=0
000018      000   C         CHECK IF THE ROOM CAN BE ASSIGNABLE
000019      000   C
000020      000   130   ICOL=IL+INI(IELT)
000021      000         DO 132 IN=1,NA
000022      000         IA=A(IN,ICOL)
000023      000         IF(IA.EQ.0) WRITE(IWT,9999)IELT,IAS,ICOL,INI(IELT),NA
000024      000         IF(BOARD(IA).NE.0) GO TO 135
000025      000   132   CONTINUE
```

```
000026      000          GO TO 136
000027      000    135   INI(IELT)=INI(IELT)+1
000028      000          IF(INI(IELT).LE.NPLOC(IELT)) GO TO 130
000029      000    C         CHANGE THE ALTERNATIVE
000030      000    C
000031      000          INI(IELT)=1
000032      000    128   IELT=IELT-1
000033      000          IF(IELT.EQ.0) GO TO 133
000034      000          COST=COST-INCRT(IELT)
000035      000    C         CLEAR THE BOARD
000036      000    C
000037      000          NA=NAREA(IELT)
000038      000          ICO=ELCOL(IELT)
000039      000          DO 129 I=1,NA
000040      000          IA=A(I,ICO)
000041      000    129   BOARD(IA)=0
000042      000          IF(INI(IELT).LT.NPLOC(IELT)) GO TO 131
000043      000          INI(IELT)=1
000044      000          GO TO 128
000045      000    131   INI(IELT)=INI(IELT)+1
000046      000          GO TO 122
000047      000    133   KOD=-1
000048      000          RETURN
000049      000    C         CALCULATE INCREMENT OF COST
000050      000    C
000051      000    136   IF(IELT.EQ.1) GO TO 139
000052      000          IAS=IELT-1
000053      000          CALL COSTA(CR,NA,IELT,ICOL)
000054      000    C         COMPARE THE INCREMENT OF COST
000055      000    C
000056      000          INCRT(IELT)=CR
000057      000          COST=COST+INCRT(IELT)
000058      000    139   ELCOL(IELT)=ICOL
000059      000    C         MAKE THE NEW ASSIGNMENT
000060      000    C
000061      000          ICO=ELCOL(IELT)
000062      000          DO 138 J=1,NA
000063      000          IA=A(J,ICO)
000064      000    138   BOARD(IA)=IELT
000065      000          IELT=IELT+1
000066      000          NA=NAREA(IELT)
000067      000          IF(IELT.LE.NELT) GO TO 122
000068      000          KOD=1
000069      000          IAS=NELT
000070      000          RETURN
000071      000          END
000072      000    @ELT,SI TUMER-S*@SC.BBOUND,,,152335102712          /
000073      000            SUBROUTINE BBOUND
000074      000    C
000075      000    C          ROUTINE CALCULATES THE NEW BOUND
000076      000    C          AND CHECKS WHETHER FURTHER IMPROVEMENT IS POSSIBLE
000077      000    C
000078      000    C          COST1 :COST BETWEEN ASSIGNED OBJECTS
000079      000    C          COST2 :COST BETWEEN UNASSIGNED AND ASSIGNED OB
000080      000    C          COST3 :COST BETWEEN UNASSIGNED OBJECTS
000081      000    C         ISAVE :SAVE AREA FOR STARTING LOCATIONS
000082      000    C
```

```
000083      000          INCLUDE BLANK
000084      000          INCLUDE INPOUT
000085      000          COST1=CSTAR
000086      000    C          REMOVE THE LAST ASSIGNED ELEMENT FROM THE BOARD
000087      000    C
000088      000          IO=3
000089      000          IELT=NELT+1
000090      000    100   IELT=IELT-1
000091      000          ICO=ELCOL(IELT)
000092      000    C     WRITE(IWT,1003) IELT,ICO
000093      000          NA=NAREA(IELT)
000094      000          NP=NPLOC(IELT)
000095      000          DO 105 I=1,NA
000096      000          IA=A(I,ICO)
000097      000    105   BOARD(IA)=0
000098      000    C     WRITE(IWT,1004)(A(I,ICO),I=1,NA)
000099      000          IF(IELT.EQ.1) IL=0
000100      000          IF(IELT.GT.1) IL=ILAST(IELT-1)
000101      000          IPOS=ICO-IL
000102      000          NOTAS(IELT,IPOS)=.TRUE.
000103      000          IAS=IAS-1
000104      000    C           COST1
000105      000    C
000106      000           COST1=COST1-INCRT(IELT)
000107      000    C         CALCULATE A NEW BOUND AFTER DLETE
000108      000    C
000109      000    101   CALL NEWBOU(COST1,BOUND,KOD)
000110      000          IF(KOD.EQ.-1) GO TO 102
000111      000          IF(BOUND.LT.CSTAR) GO TO 104
000112      000          GO TO 102
000113      000    C         CHECK WHETHER IT IS PHYSICALLY POSSIBLE
000114      000    C         TO ASSIGN THE DELETED ELEMENT SOMEWHERE ELSE
000115      000    C
000116      000    104   DO 106 I=1,NP
000117      000          ICOL=IL+I
000118      000          DO 107 II=1,NA
000119      000          IA=A(II,ICOL)
000120      000          IF(BOARD(IA).NE.0.OR.NOTAS(IELT,I)) GO TO 106
000121      000          IPOS=I
000122      000    107   CONTINUE
000123      000    C         ASSIGN THE ELEMENT TO ITS NEW POSITION
000124      000    C
000125      000          CALL COSTA(CR,NA,IELT,ICOL)
000126      000          INCRT(IELT)=CR
000127      000          COST1=COST1+CR
000128      000          ELCOL(IELT)=ICOL
000129      000    C     WRITE(IWT,1005) IELT,I,CR
000130      000          DO 108 IT=1,NA
000131      000          IA=A(IT,ICOL)
000132      000    108   BOARD(IA)=IELT
000133      000    C     WRITE(IWT,1004)(A(IT,ICOL),IT=1,NA)
000134      000          IAS=IAS+1
000135      000    C         END OF THE ELEMENTS
000136      000    C
000137      000          IF(IAS.LT.NELT) GO TO 110
000138      000          IF(CSTAR.GT.COST1) GO TO 109
000139      000    C     WRITE(IWT,1002)COST1,CSTAR
```

```
000140  000          IELT=NELT+1
000141  000          GO TO 100
000142  000    109   CSTAR=COST1
000143  000          CALL OUTPUT(IO)
000144  000          IO=0
000145  000          IELT=NELT+1
000146  000          GO TO 100
000147  000    106   CONTINUE
000148  000    C         FREE THE LOCATIONS
000149  000    C
000150  000    102   IF(IELT.EQ.1.AND.IPOS.EQ.ILAST(1)) RETURN
000151  000          IF(IELT.GT.1) GO TO 111
000152  000          NOTAS(1,IPOS)=.TRUE.
000153  000          GO TO 104
000154  000    111   DO 103 I=1,NP
000155  000    103   NOTAS(IELT,I)=.FALSE.
000156  000          GO TO 100
000157  000    C
000158  000    C         CHECK FOR A BETTER BOUND
000159  000    C
000160  000    110   CALL NEWBOU(COST1,BOUND,KOD)
000161  000          IF(KOD.EQ.-1) GO TO 112
000162  000           IF(BOUND.GE.CSTAR) GO TO 112
000163  000    C           TRY TO ASSIGN THE NEXT ELEMENT
000164  000    C
000165  000          IELT=IELT+1
000166  000          NP=NPLOC(IELT)
000167  000          NA=NAREA(IELT)
000168  000          IL=ILAST(IELT-1)
000169  000          GO TO 104
000170  000    C112   WRITE(IWT,1006) IELT,IPOS
000171  000    112   NOTAS(IELT,IPOS)=.TRUE.
000172  000          DO 113 IT=1,NA
000173  000          IA=A(IT,ICOL)
000174  000    113   BOARD(IA)=0
000175  000    C     WRITE(IWT,1004) (A(IT,ICOL),IT=1,NA)
000176  000          COST1=COST1-INCRT(IELT)
000177  000          IAS=IAS-1
000178  000          GO TO 104
000179  000    1002  FORMAT(/,2X,'NEW COST=',F12.2,' OPTIMUM COST=',F12.2)
000180  000    1003  FORMAT(/2X,'ROOM ',I3,' IS DELETED FROM POSITION:',I4,
000181  000         */2X,'BLOCKS:')
000182  000    1004  FORMAT(2X,20I5)
000183  000    1005  FORMAT(/2X,'ROOM',I4,' IS ASSIGNED TO POSITION',I4,
000184  000         *'  WITH COST:',F10.2/2X,'BLOCK')
000185  000    1006  FORMAT(/2X,'ASSIGN ROOM',I4,' PRESENTLY IN POSITION',I4,
000186  000         *' ,TO NEXT FREE POSITION',/2X,'BLOCKS',/)
000187  000                END
000188  000    @ELT,SI TUMER-S*@SC.COSTA,,,141302101112
000189  000          SUBROUTINE COSTA(CR,NA,IELT,ICOL)
000190  000    C         INCREMENTAL COST BETWEEN ASSIGNED ELEMENTS
000191  000          INCLUDE BLANK
000192  000          CR=0.
000193  000          R2=NA
000194  000          DO 133 IIE=1,IAS
000195  000          R1=NAREA(IIE)
000196  000          R3=IWEIG(IELT,IIE)
```

```
000197      000   C          FIND UNIT COMMUNICATION AND INCREMENT OF COST
000198      000   C
000199      000         ICO=ELCOL(IIE)
000200      000         UNIT=R3/(R1*R2)
000201      000         INA=NAREA(IIE)
000202      000         DO 134 JE=1,INA
000203      000         I1=A(JE,ICO)
000204      000         DO 134 JM=1,NA
000205      000         I2=A(JM,ICOL)
000206      000   134   CR=CR+IDIST(I1,I2)*UNIT
000207      000   133   CONTINUE
000208      000         RETURN
000209      000         END
000210      000   @ELT,SI TUMER-S*QSC.COSTUN,,,[illegible]101112
000211      000         SUBROUTINE COSTUN(COST,NA,IUNAS,ICOL,KOD)
000212      000   C
000213      000   C          ROUTINE CALCULATES THE COST
000214      000   C          BETWEEN UNASSIGNED ELEMENTS
000215      000   C
000216      000         INCLUDE BLANK
000217      000         INCLUDE INPOUT
000218      000         BIG=1.0E38
000219      000         IBEG=IUNAS+1
000220      000         R2=NA
000221      000         COST=0.
000222      000         DO 130 IEL=IBEG,NELT
000223      000         KOD=-1
000224      000         SAVE=BIG
000225      000         INA=NAREA(IEL)
000226      000         NP=NPLOC(IEL)
000227      000         IL=ILAST(IEL-1)
000228      000         R1=INA
000229      000         R3=IWEIG(IUNAS,IEL)
000230      000         UNIT=R3/(R1*R2)
000231      000         DO 125 ILOC=1,NP
000232      000         ICO=ILOC+IL
000233      000   C          CHECK WHETHER ASSIGNMENT CAN BE MADE
000234      000   C
000235      000         DO 120 IAREA=1,INA
000236      000         IA=A(IAREA,ICO)
000237      000         DO 121 I=1,NA
000238      000         IF(IA.EQ.A(I,ICOL)) GO TO 125
000239      000   121    CONTINUE
000240      000         IF(IA.EQ.0) WRITE(IWT,9999)IUNAS,IEL,ICOL,ICO,NA,INA
000241      000         IF(BOARD(IA).NE.0.OR.NOTAS(IEL,ILOC)) GO TO 125
000242      000   120   CONTINUE
000243      000         KOD=1
000244      000         CR=0.
000245      000   C          CALCULATE COST BETWEEN OBJECTS
000246      000   C
000247      000         DO 134 JE=1,INA
000248      000         DO 134 JM=1,NA
000249      000         I1=A(JE,ICO)
000250      000         I2=A(JM,ICOL)
000251      000   134   CR=CR+IDIST(I1,I2)*UNIT
000252      000         IF(SAVE.GT.CR) SAVE=CR
000253      000   125   CONTINUE
```

```
000254  000          IF(KOD.EQ.-1) RETURN
000255  000          COST=COST+SAVE
000256  000    130   CONTINUE
000257  000          RETURN
000258  000          END
000259  000    @ELT,SI TUMER-S*QSC.DISTCE,,,113117062712
000260  000          SUBROUTINE DISTCE(KOD)
000261  000          INCLUDE BLANK
000262  000          INCLUDE INPOUT
000263  000    C         ROUTINE WHICH CALCULATES IDISTANCES ON THE BOARD
000264  000          INTEGER X(IBLOCK),Y(IBLOCK)
000265  000          N=NBLK
000266  000          READ(IRD,1000) (X(I),Y(I),I=1,N)
000267  000    1000  FORMAT(16I5)
000268  000          N1=N-1
000269  000          GO TO (100,200,300),KOD
000270  000    C         RECTILINEAR IDISTANCES KOD=1
000271  000    C
000272  000    100   DO 110 I=1,N1
000273  000          L=I+1
000274  000          DO 110 J=L,N
000275  000          IDIST(I,J)=IABS(X(I)-X(J))+IABS(Y(I)-Y(J))
000276  000    110   IDIST(J,I)=IDIST(I,J)
000277  000          GO TO 999
000278  000    C         EUCLIDEAN IDISTANCES KOD=2
000279  000    C
000280  000    200   DO 210 I=1,N1
000281  000          L=I+1
000282  000          DO 210 J=L,N
000283  000          IDIST(I,J)=SQRT((X(I)-X(J))**2+(Y(I)-Y(J))**2)
000284  000    210   IDIST(J,I)=IDIST(I,J)
000285  000          GO TO 999
000286  000    C         EUCLIDEAN SQUARE IDISTANCE
000287  000    C
000288  000    300   DO 310 I=1,N1
000289  000          L=I+1
000290  000          DO 310 J=L,N
000291  000          IDIST(I,J)=(X(I)-X(J))**2+(Y(I)-Y(J))**2
000292  000    310   IDIST(J,I)=IDIST(I,J)
000293  000    C         OUTPUT DISTANCE MATRIX
000294  000    C
000295  000    C999   WRITE(IWT,1001)
000296  000    C     DO 998 I=1,N
000297  000    C998   WRITE(IWT,1000) (IDIST(I,J),J=1,N)
000298  000    C1001  FORMAT(//6X,'DISTANCE MATRIX',//)
000299  000    999   RETURN
000300  000          END
000301  000    @ELT,SI TUMER-S*QSC.FORS,,,103724102312
000302  000    @FOR,S QSC.ASSGN
000303  000    @FOR,S QSC.COSTA
000304  000    @FOR,S QSC.DISTCE
000305  000    @FOR,S QSC.INASS
000306  000    @FOR,S QSC.INPUT
000307  000    @FOR,S QSC.NEWBOU
000308  000    @FOR,S QSC.MAIN
000309  000    @FOR,S QSC.OUTPUT
000310  000    @FOR,S QSC.TIME
```

```
000311      000   @FOR,S QSC.BBOUND
000312      000   @FOR,S QSC.COSTUN
000313      000   @EOF
000314      000   @ELT,SI TUMER-S*QSC.FORY,,,1375241011112
000315      000   @FOR,Y QSC.ASSGN
000316      000   @FOR,Y QSC.COSTA
000317      000   @FOR,Y QSC.DISTCE
000318      000   @FOR,Y QSC.INASS
000319      000   @FOR,Y QSC.INPUT
000320      000   @FOR,Y QSC.NEWROU
000321      000   @FOR,Y QSC.MAIN
000322      000   @FOR,Y QSC.OUTPUT
000323      000   @FOR,Y QSC.TIME
000324      000   @FOR,Y QSC.BROUND
000325      000   @FOR,Y QSC.COSTUN
000326      000   @EOF
000327      000   @ELT,SI TUMER-S*QSC.INASS,,,1375241012612
000328      000         SUBROUTINE INASS(IBAR)
000329      000   C         MAKE THE INITIAL ASSIGNMENT ON THE BOARD
000330      000         INCLUDE BLANK
000331      000         INCLUDE INPOUT
000332      000         IO=1
000333      000         ILAST(1)=NPLOC(1)
000334      000         DO 117 I=2,NELT
000335      000   117   ILAST(I)=ILAST(I-1)+NPLOC(I)
000336      000         CSTAR=1.0E38
000337      000         DO 119 I=1,NELT
000338      000   119   INI(I)=1
000339      000         IAS=0
000340      000   C         ASSIGN THE OBJECTS
000341      000   C
000342      000         DO 122 I=1,NBLK
000343      000   122   BOARD(I)=0
000344      000         COST=0.
000345      000   121   CALL ASSGN(KOD,COST)
000346      000   C          COMPARE THE PREVIOUS COST FOR IMPROVEMENT
000347      000   C
000348      000         IF(KOD.EQ.-1)  GO TO 994
000349      000         MI=MI+1
000350      000         IF(CSTAR.LE.COST) GO TO 140
000351      000         CSTAR=COST
000352      000   C         OUTPUT THE ASSIGNMENTS
000353      000   C
000354      000   145   CALL OUTPUT(IO)
000355      000         IF(IO.EQ.1) IO=2
000356      000   C         CHECK TIME FOR FURTHER IMPROVEMENT
000357      000   C
000358      000   140   IT1=ITIME(IT2,IT3)
000359      000         T=IT2/5000.
000360      000         IF(T.GT.MAXTIM) RETURN
000361      000   C         SEARCH FOR BETTER INITIAL SOLUTION
000362      000   C
000363      000   155   INI(IAS)=INI(IAS)+1
000364      000         COST=COST-INCRT(IAS)
000365      000         NA=NAREA(IAS)
000366      000         ICO=ELCOL(IAS)
000367      000         DO 150 I=1,NA
```

```
000368      000          IA=A(I,ICO)
000369      000    150   BOARD(IA)=0
000370      000          IAS=IAS-1
000371      000          IF(IAS.LT.0) GO TO 994
000372      000          IF(INI(IAS+1).LE.NPLOC(IAS+1)) GO TO 121
000373      000          INI(IAS+1)=1
000374      000          GO TO 155
000375      000    994   MI=MI+1
000376      000          IF(MI.EQ.1) GO TO 995
000377      000          WRITE(IWT,101)
000378      000          IBAB=1
000379      000          RETURN
000380      000    995   WRITE(IWT,100)
000381      000          STOP
000382      000    100   FORMAT(//' STOP** THERE IS NO POSSIBILITY TO MAKE COMPLETE '
000383      000         *,'INITIAL ASSIGNMENT')
000384      000    101   FORMAT(//' IT IS NOT POSSIBLE TO MAKE FURTHER ',
000385      000         *'IMPROVEMENT')
000386      000          END
000387      000    @ELT,SI TUMER-S*QSC.INPUT,,,197686071012
000388      000          SUBROUTINE INPUT
000389      000    C         IDIS  :=1 DISTANCE WILL BE CALCULATED
000390      000    C               =0 DISTANCE WILL BE READ
000391      000          INCLUDE BLANK
000392      000          INCLUDE INPOUT
000393      000    C         READ DATA
000394      000    C
000395      000          READ(IRD,1001) NELT,NBLK,MAXTIM,IDIS
000396      000    C         ELEMENT CARDS
000397      000    C
000398      000          DO 104 I=1,NELT
000399      000          READ(IRD,1002) M,NPLOC(M),NAREA(M),NLIM(M)
000400      000          IF(I.NE.M) GO TO 997
000401      000    104   CONTINUE
000402      000    C         LIMITATION CARDS
000403      000    C
000404      000          DO 102 IE=1,NELT
000405      000          NL=NLIM(IE)
000406      000          IF(NL.EQ.0) GO TO 102
000407      000          READ(IRD,1001)M,(LIMIT(M,I),I=1,NL)
000408      000          IF(IE.NE.M.OR.LIMIT(M,NL).EQ.0) GO TO 998
000409      000    102   CONTINUE
000410      000    C         CREATE A MATRIX
000411      000          IC=0
000412      000          DO 110 I=1,NELT
000413      000          NP=NPLOC(I)
000414      000          NA=NAREA(I)
000415      000          DO 110 J=1,NP
000416      000          IC=IC+1
000417      000          READ(IRD,1001) M,(A(II,IC),II=1,NA)
000418      000          IF(I.NE.M.OR.A(NA,IC).EQ.0.OR.A(NA+1,IC).NE.0) GO TO 999
000419      000    110   CONTINUE
000420      000    C         READ DISTANCES BETWEEN EACH BLOCK
000421      000    C
000422      000          IF(IDIS.EQ.0) GO TO 113
000423      000          CALL DISTCE(1)
000424      000          GO TO 114
```

```
000425      000
000426      000     113   NBL=NBLK-1
000427      000           DO 111 I=1,NBL
000428      000           K=I+1
000429      000           READ(IRD,1001) M,(IDIST(M,MJ),MJ=K,NBLK)
000430      000           IF(I.NE.M) GO TO 995
000431      000           DO 111 J=1,NBLK
000432      000     111   IDIST(J,I)=IDIST(I,J)
000433      000     C         READ WEIGHT BETWEEN ELEMENTS
000434      000     C
000435      000     114   NLT=NELT-1
000436      000           DO 112 I=1,NLT
000437      000           K=I+1
000438      000           READ(IRD,1001) M,(IWEIG(M,MJ),MJ=K,NELT)
000439      000           IF(I.NE.M) GO TO 996
000440      000           DO 112 J=1,NELT
000441      000     112   IWEIG(J,I)=IWEIG(I,J)
000442      000           RETURN
000443      000     C         ERROR MESSAGES
000444      000     C
000445      000     995   WRITE(IWT,1006) I,M
000446      000           STOP
000447      000     996   WRITE(IWT,1007) I,M
000448      000           STOP
000449      000     997   WRITE(IWT,1003) I,M
000450      000           STOP
000451      000     998   WRITE(IWT,1004) IE,M
000452      000           STOP
000453      000     999   WRITE(IWT,1005) I,M
000454      000           STOP
000455      000     C         FORMAT STATEMENTS
000456      000     1001  FORMAT(16I5)
000457      000     1002  FORMAT(4I5)
000458      000     1003  FORMAT(//'  STOP** CHECK ELEMENT CARD:',I5,2X,'M=',I5)
000459      000     1004  FORMAT(//'  STOP** CHECK LOCATION CARD:',2X,'M=',I5)
000460      000     1005  FORMAT(//'  STOP** CHECK ELEMENT:',I5,' OF BLOCK A  M=',I5)
000461      000     1006  FORMAT(//'  STOP** CHECK DISTANCE MATRIX ELEMENT:',I5,'  M=',I5)
000462      000     1007  FORMAT(//'  STOP** CHECK WEIGHT MATRIX ELEMENT:',I5,'  M=',I5)
000463      000           END
000464      000     @ELT,SI TUMER-S*QSC.MAIN,,,15,72,01112
000465      000     C
000466      000     C
000467      000     C         Q U A D R A T I C   S E T   C O V E R I N G   P R O B L E M
000468      000     C
000469      000     C
000470      000     C         VARIABLES:
000471      000     C
000472      000     C         A      :BINARY MATRIX WHICH SHOWS POSSIBLE LOCATIONS OF ELEMENTS
000473      000     C         ASRANK:ASSIGNMENT ORDER
000474      000     C         AUXBOA:AUXILIARY ASSIGNMENT BOARD FOR MODIFICATION
000475      000     C         BOARD :ASSIGNMENT BOARD
000476      000     C         CSTAR :OPTIMAL COST (C1*C2*C3)
000477      000     C         ELCOL :THE BEST CHOICE OF LOCATION OF ELEMENT I
000478      000     C         IAS   :NUMBER OF ALREADY ASSIGNED ELEMENTS
000479      000     C         ICOL  :COLUMN NUMBER OF MATRIX A
000480      000     C         IDIST :DISTANCE FROM BLOCK I TO BLOCK J
000481      000     C         IELT  :ELEMENT NUMBER
```

```
000482  000   C          ILAST :LAST COLUMN OF POSSIBLE LOCATIONS OF EACH ELEMENT IN A MATRIX
000483  000   C          INI   :INITIAL BEGINNING COLUMN FOR TRIALS
000484  000   C           INCRT :INCREMENT OF COST AFTER ASSIGNMENT
000485  000   C          IUNIT :UNIT COMMUNICATION BETWEEN THE BLOCKS OF YHE ELEMENTS
000486  000   C          IWEIG :WEIGHT BETWEEN ELEMENTS
000487  000   C          IFIRST:ELEMENT WHICH THE FIRST ASSIGNMENT WAS DONE
000488  000   C          LIMIT :ADJACENT ASSIGNMENT LIMITATION
000489  000   C          MAXTIM:MAXIMUM TIME IN SECONDS TILL THE INITIAL SOLUTION
000490  000   C                 WILL BE IMPROVED
000491  000   C          NAREA :NUMBER OF BLOCKS WHICH AN ELEMENT OCCUPIES
000492  000   C          NBLK  :NUMBER OF BLOCKS
000493  000   C          NELT  :NUMBER OF ELEMENTS
000494  000   C          NLIM  :NUMBER OF LIMITATAIONS ON EACH ELEMENT
000495  000   C          NPLOC :NUMBER OF POSSIBLE LOCATIONS WHERE AN ELEMENT CAN BE LOCATED
000496  000   C
000497  000         INCLUDE BLANK
000498  000         INCLUDE INPOUT
000499  000   C
000500  000   C          READ DATA
000501  000         CALL INPUT
000502  000   C          MAKE THE INITIAL ASSIGNMENT
000503  000         CALL INASS(IBAB)
000504  000         IF(IBAB.EQ.1) STOP
000505  000         CALL BBOUND
000506  000         END
000507  000   @FLT,SI TUMER-S*QSC.MAPN-XQT,,,140054101112
000508  000   @MAP,N
000509  000   IN QSC.MAIN
000510  000   IN QSC.INPUT
000511  000   IN QSC.INASS
000512  000   IN QSC.COSTA
000513  000   IN QSC.NEWBOU
000514  000   IN QSC.ASSGN
000515  000   IN QSC.DISTCE
000516  000   IN QSC.OUTPUT
000517  000   IN QSC.TIME
000518  000   IN QSC.BBOUND
000519  000   IN QSC.COSTUN
000520  000   LIB SYSTEM$*MATHSTAT.
000521  000   END
000522  000   @XQT
000523  000   @ELT,SI TUMER-S*QSC.MAPS-XQT,,,104266102312
000524  000   @MAP,S
000525  000   IN QSC.MAIN
000526  000   IN QSC.INPUT
000527  000   IN QSC.INASS
000528  000   IN QSC.COSTA
000529  000   IN QSC.NEWBOU
000530  000   IN QSC.ASSGN
000531  000   IN QSC.DISTCE
000532  000   IN QSC.OUTPUT
000533  000   IN QSC.TIME
000534  000   IN QSC.BBOUND
000535  000   IN QSC.COSTUN
000536  000   LIB SYSTEM$*MATHSTAT.
000537  000   END
000538  000   @XQT
```

```
000539   000   @ELT,SI TUNER-S*QSC.NEWBOU,,,1087?5102712
000540   000         SUBROUTINE NEWBOU(COST1,BOUND,KOD)
000541   000   C++ROUTINE CALCULATES THE NEW BOUND
000542   000   C
000543   000         INCLUDE BLANK
000544   000         INCLUDE INPOUT
000545   000         BIG=1.E38
000546   000         COST23=0.
000547   000         IB=IAS+1
000548   000          DO 130 IUNAS=IB,NELT
000549   000         KOD1=-1
000550   000         KOD2=-1
000551   000         SAVE=BIG
000552   000          NA=NAREA(IUNAS)
000553   000          NP=NPLOC(IUNAS)
000554   000          IF(IUNAS.EQ.1) IL=0
000555   000          IF(IUNAS.GT.1) IL=ILAST(IUNAS-1)
000556   000          DO 125 ILOC=1,NP
000557   000          ICOL=ILOC+IL
000558   000   C            CALCULATE COST2 & COST3
000559   000   C
000560   000         DO 120 IAREA=1,NA
000561   000         IA=A(IAREA,ICOL)
000562   000         IF(IA.EQ.0) WRITE(IWT,9999) IUNAS,IAS
000563   000   C            CHECK WHETHER ASSIGNMENT CAN BE MADE
000564   000   C
000565   000         IF(BOARD(IA).NE.0.OR.NOTAS(IUNAS,ILOC))GO TO 125
000566   000   120   CONTINUE
000567   000         KOD2=1
000568   000         CR=0.
000569   000         CU=0.
000570   000   C            COST2
000571   000   C
000572   000         IF(IAS.EQ.0) GO TO 121
000573   000          CALL COSTA(CR,NA,IUNAS,ICOL)
000574   000   C            COST3
000575   000   C
000576   000   121   IF(IUNAS.EQ.NELT) GO TO 124
000577   000          CALL COSTUN(CU,NA,IUNAS,ICOL,KOD)
000578   000          IF(KOD.EQ.-1) GO TO 125
000579   000   124   C23=CR+CU
000580   000         IF(SAVE.GT.C23) SAVE=C23
000581   000         KOD2=1
000582   000   125    CONTINUE
000583   000         IF(KOD1.EQ.-1.OR.KOD2.EQ.-1) GO TO 102
000584   000   C
000585   000         COST23=COST23+SAVE
000586   000   130   CONTINUE
000587   000         BOUND=COST1+COST23
000588   000   C     WRITE(IWT,1000) IAS
000589   000   C     WRITE(IWT,1001) BOUND,COST1,COST2,COST3
000590   000   C     CALL TIME
000591   000         RETURN
000592   000   C102   WRITE(IWT,1002) IAS
000593   000   102   KOD=-1
000594   000         RETURN
000595   000   1000  FORMAT(/,2X,'FOR ',I2,' ASSIGNED ELEMENTS BRANCH AND',
```

```
000596  000        *' BOUND RESULTS:')
000597  000   1001  FORMAT(/2X,'NEWBOUND=',F11.2,' COST1=',F11.2,' COST2=',F11.2,
000598  000        *,' COST3=',F11.2)
000599  000   1002  FORMAT(/2X,'FOR',I4,' ASSIGNED ELEMENTS BRANCH AND BOUND',
000600  000        *' PHYSICALLY IS NOT POSSIBLE')
000601  000        END
000602  000   @ELT,SI TUMER-S*QSC.OUTPUT,,,152177102712
000603  000        SUBROUTINE OUTPUT(M)
000604  000   C         OUTPUT THE RESULTS
000605  000        INCLUDE BLANK
000606  000        INCLUDE INPOUT
000607  000        IF(M.EQ.0) GO  TO 110
000608  000        IF(M.EQ.1) WRITE(IWT,1010)
000609  000        IF(M.EQ.2) WRITE(IWT,1012) MAXTIM
000610  000        IF(M.EQ.3)WRITE(IWT,1013)
000611  000        IF(M.EQ.2) M=0
000612  000   110  WRITE(IWT,1009) CSTAR
000613  000        WRITE(IWT,1011)
000614  000        DO 149 I=1,NELT
000615  000        IA=ELCOL(I)
000616  000        NA=NAREA(I)
000617  000   149  WRITE(IWT,1001) I,(AIJ,IA),J=1,NA)
000618  000        CALL TIME
000619  000        RETURN
000620  000   1001 FORMAT(1615)
000621  000   1009 FORMAT(//2X,'*** COST=',F12.2)
000622  000   1010 FORMAT(////2X,'INITIAL ASSIGNMENTS'/2X,19('-'))
000623  000   1011 FORMAT(/2X,'ROOM',2X,'LOCATIONS ON THE BOARD ----------')
000624  000   1012 FORMAT(////2X,'IMPROVED INITIAL ASSIGNMENTS'/2X,27('-')/
000625  000       *2X,'IMPROVEMENT TILL',I5,' SECONDS')
000626  000   1013 FORMAT(///2X,'BRANCH AND BOUND IMPROVEMENT'/2X,29('-'))
000627  000        END
000628  000   @ELT,SI TUMER-S*QSC.RFOR,,,137545101112
000629  000   @RFOR,CS QSC.ASSGN
000630  000   @RFOR,CS QSC.COSTA
000631  000   @RFOR,CS QSC.DISTCE
000632  000   @RFOR,CS QSC.INASS
000633  000   @RFOR,CS QSC.INPUT
000634  000   @RFOR,CS QSC.NEWBOU
000635  000   @RFOR,CS QSC.MAIN
000636  000   @RFOR,CS QSC.OUTPUT
000637  000   @RFOR,CS QSC.TIME
000638  000   @RFOR,CS QSC.BBOUND
000639  000   @RFOR,CS QSC.COSTUN
000640  000   @EOF
000641  000   @ELT,SI TUMER-S*QSC.TIME,,,157000061312
000642  000        SUBROUTINE TIME
000643  000        INCLUDE INPOUT
000644  000        K=ITIME(I,J)
000645  000        T=I/5000.
000646  000        WRITE(IWT,2000)T
000647  000    2000 FORMAT(' TIME',F10.3,' SEC')
000648  000        RETURN
000649  000        END
000650  000   @PDP,FI TUMER-S*QSC.BLANK,,,154360101112
000651  000   BLANK   PROC
000652  000        PARAMETER IROOM=25,IBLOCK=80
```

```
000653      000          PARAMETER ICLMN=200,IROWS=30
000654      000          COMMON NPLOC(IROOM),NAREA(IROOM),LIMIT(IROOM,
000655      000         *NLIN(IROOM),IWEIG(IROOM,
000656      000         *ILAST(IROOM),A(IROWS,ICLMN),ASRANK(IROOM),NOTAS(IROOM,
000657      000         *IROOM),INCRT(IROOM),INI
000658      000         *,ELCOL(IROOM),IAS,MAXTIM,
000659      000         *CSTAR
000660      000          INTEGER A,ASRANK,ELCOL,BOARD
000661      000          REAL INCRT
000662      000          LOGICAL NOTAS
000663      000       END
000664      000    @PDP,FI TUMER-S*@SC.INPOUT,,,154402101112
000665      000    INPOUT  PROC
000666      000          PARAMETER IWT=6,IRD=5
000667      000       9999 FORMAT( )
000668      000       END
000669      000    @PDP,FI TUMER-S*@SC.BLANK,,,154402101112
000670      000    BLANK   PROC
000671      000          PARAMETER IROOM=25,IBLOCK=50
000672      000          PARAMETER ICLMN=200,IROWS=30
000673      000          COMMON NPLOC(IROOM),NAREA(IROOM),LIMIT(IROOM,
000674      000         *NLIN(IROOM),IWEIG(IROOM,
000675      000         *ILAST(IROOM),A(IROWS,ICLMN),ASRANK(IROOM),NOTAS(IROOM,
000676      000         *IROOM),INCRT(IROOM),INI
000677      000         *,ELCOL(IROOM),IAS,MAXTIM,
000678      000         *CSTAR
000679      000          INTEGER A,ASRANK,ELCOL,BOARD
000680      000          REAL INCRT
000681      000          LOGICAL NOTAS
000682      000       END
000683      000    @PDP,FI TUMER-S*@SC.INPOUT,,,154402101112
000684      000    INPOUT  PROC
000685      000          PARAMETER IWT=6,IRD=5
000686      000       9999 FORMAT( )
000687      000       END
000688      000
000689      000    RUNID IE0023 FILE PU@000IE0023 PART NO 000 DATE 051779 UNIT CP1    IE0023
000690      000    @ELT,SI TUMER-S*SINAN-BAB.BBOUND,,,170256041312
000691      000    C
000692      000    C
000693      000    C
000694      000    C          B R A N C H   A N D   B O U N D
000695      000    C
000696      000    C
000697      000    C          VARIABLES:
000698      000    C          ASELT :LOCATION OF THE ASSIGNED ELEMENT
000699      000    C          ASRANK :ASSIGNMENT ORDER OF THE ELEMENTS
000700      000    C          ASSEL :ASSIGNED ELEMENTS
000701      000    C          AUX-I :AREAS USED FOR TEMPORARY CALCULATIONS
000702      000    C          AUXBOA:AUXILIARY ASSIGNMENT BOARD FOR MODIFICATION
000703      000    C          BOARD :ASSIGNMENT BOARD
000704      000    C          DISTFR:LOCATION WHICH THE DISTANCE MEASURED FROM
000705      000    C          DISTTO:LOCATION WHICH THE DISTANCE MEASURED TO
000706      000    C          ELEFR :ELEMENT WHICH THE CONNECTION LEAVES (FROM)
000707      000    C          ELETO :ELEMENT TO WHICH THE CONNECTION REACHES
000708      000    C          ELTCON:NO. OF CONNECTIONS FROM ELEMENT I TO ELEMENT J
000709      000    C          FINRAN:ELEMENTS IN DESCENDING ORDER ACCORDING TO
```

```
000710      000    C          IAS   :ASSIGNED ELEMENT NUMBER
000711      000    C          ICOL  :ELEMENT NUMBER TO WHICH CONNECTION ENTERS
000712      000    C          IDIST :DISTANCE FROM LOCATION I TO LOCATION J
000713      000    C          IDISTS:LOCATIONS IN ASCENDING ORDER ACCORDING
000714      000    C                 TO TOTAL DISTANCES FROM EACH LOCATION
000715      000    C          IELTC3:ARRAY WHICH THE ELEMENT CONNECTIONS ARE SORTED
000716      000    C          ILOCC3:ARRAY WHICH LOCATIONS OF DISTANCES ARE HELD
000717      000    C          IPERC :PERCENT OF ACCURACY DESIRED ON OPTIMUM SOLUTION
000718      000    C          ILO   :LOCATION NUMBER WHICH THE ELEMENT HAS BEEN ASSIGNED
000719      000    C          IRE   :ELEMENT NUMBER WHICH IS REMOVED
000720      000    C          IROW  :ELEMENT NUMBER FROM WHICH CONNECTION LEAVES
000721      000    C          ITREE :BOARD WHICH SHOWS THE REMOVED ELEMENTS FROM TREE
000722      000    C          KCON  :TOTAL NUMBER OF ELEMENT CONNECTIONS
000723      000    C          KDIS  :TOTAL NUMBER OF DISTANCES BETWEEN LOCATIONS
000724      000    C          KODIS :=0 IF RECTILINEAR DISTANCES ARE DESIRED
000725      000    C                 =1 IF EUCLIDEAN DISTANCES ARE DESIRED
000726      000    C          LOCOD :=0 IF DISTANCES ARE TO BE CALCULATED BY THE
000727      000    C                 PROGRAM , OR THE ASSIGNMENT SHEET
000728      000    C                 =1 IF DISTANCES ARE TO BE READ EXTERNALLY
000729      000    C          NCONS :ELEMENTS IN ASCENDING ORDER ACCORING
000730      000    C                 TO NUMBER OF CONNECTIONS TO OTHER ELEMENTS
000731      000    C          MELT  :ASSIGNMENT ORDER
000732      000    C          NASSEL:NUMBER OF ASSIGNED ELEMENTS
000733      000    C          NDIM  :MINIMUM DIMENSIONS OF ASSIGNMENT BOARD
000734      000    C          NELT  :NUMBER OF ELEMENTS
000735      000    C          NLOC  :NUMBER OF LOCATIONS
000736      000    C          NWIRES:ELEMENTS IN ASCENDING ORDER ACCORDING
000737      000    C                 TO TOTAL NUMBER OF WIRES
000738      000    C
000739      000    C
000740      000          COMMON/AUXILI/AUX5(800),AUX6(800)
000741      000          COMMON/AREA/AUXBOA(40),AUX1(40),AUX2(40),AUX3(40),ITREE(35,40),
000742      000         1IDIST(40,40),ELTCON(35,35),ASELT(35),ASSEL(35),NELT,NLOC,NASSEL
000743      000         2,KDIS,KCON,IELTC3(650),ILOCC3(800),ELEFR(650),ELTO(650),
000744      000         3DISTFR(800),DISTTO(800)
000745      000          DIMENSION ICOL(5),IVAL(5),IDISTS(40),NWIRES(35),NCONS(35)
000746      000          INTEGER ELTCON,BOARD(40),EMPTY,FINBAN(35),AUX1,AUX2,AUX3,
000747      000         1ASRANK(35),ASELT,AUXBOA,FROM,TO,CSTAR,CNEW,C1,C2,C3,ASSEL
000748      000         2,ELETO,ELEFR,DISTFR,DISTTO,AUX4(800),AUX5,AUX6,LOCER
000749      000          IFIN=1
000750      000          IRD=5
000751      000          IWT=6
000752      000          INFIN=10E+10
000753      000          EMPTY=0
000754      000          READ(IRD,1001) NELT,IPERC,IC2,LOCOD,KODIS
000755      000          IF(LOCOD.EQ.0) GO TO 100
000756      000    C         READ DISTANCES FROM EACH LOCATION
000757      000    C
000758      000          READ(IRD,1001) NLOC
000759      000          NLC=NLOC-1
000760      000          DO 96 I=1,NLC
000761      000          K=I+1
000762      000          READ(IRD,1003) (IDIST(I,J),J=K,NLOC)
000763      000          DO 96 J=1,NLOC
000764      000    96    IDIST(J,I)=IDIST(I,J)
000765      000    C         READ NUMBER OF CONNECTION BETWEEN EACH ELEMENT
000766      000    C
```

```
000767      000   100  READ(IRD,1002,END=101) IROW,(ICOL(M),IVAL(M),M=1,5)
000768      000        IF(IROW.EQ.0) GO TO 103
000769      000        IR=IROW
000770      000   103  CONTINUE
000771      000   C        CREATE THE MATRIX OF CONNECTIONS
000772      000   C
000773      000        DO 102 M=1,5
000774      000        IF(ICOL(M).EQ.0) GO TO 100
000775      000        IC=ICOL(M)
000776      000        ELTCON(IR,IC)=IVAL(M)
000777      000        ELTCON(IC,IR)=IVAL(M)
000778      000   102  CONTINUE
000779      000        GO TO 100
000780      000   101    IF(LOCOD.EQ.1) GO TO 110
000781      000   C        CALCULATE ALL DISTANCES ON THE ASSIGNMENT BOARD AND SUM
000782      000   C        THE DISTANCES FROM ONE LOCATION TO ALL OTHER LOCATIONS
000783      000   C
000784      000        XELT=NELT
000785      000        NDIM=SQRT(XELT)+1
000786      000        NLOC=NDIM*NDIM
000787      000        I=0
000788      000        DO 106 ILOC=1,NLOC
000789      000        AUX1(ILOC)=0
000790      000        IF(ILOC.LE.I*NDIM) GO TO 104
000791      000        I=I+1
000792      000        J=0
000793      000   104  J=J+1
000794      000        TO=0
000795      000        FROM=ILOC
000796      000        DO 106 M=1,NDIM
000797      000        DO 106 K=1,NDIM
000798      000        TO=TO+1
000799      000        J1=IABS(J-K)
000800      000        I1=IABS(I-M)
000801      000        IF(KODIS.EQ.0) IDIST(FROM,TO)=I1+J1
000802      000        IF(KODIS.EQ.1) IDIST(FROM,TO)=(I1*I1+J1*J1)**0.5
000803      000   106  AUX1(ILOC)=AUX1(ILOC)+IDIST(FROM,TO)
000804      000        GO TO 109
000805      000   C        SUM DISTANCES FROM EACH LOCATION (READ EXPLICITELY)
000806      000   C
000807      000   110  DO 105 FROM=1,NLOC
000808      000        AUX1(FROM)=0
000809      000        DO 105 TO=1,NLOC
000810      000   105  AUX1(FROM)=AUX1(FROM)+IDIST(FROM,TO)
000811      000   C        SORT SUM OF DISTANCES IN ASCENDING ORDER
000812      000   C
000813      000   109  CALL SORT(AUX1,IDISTS,NLOC,NLOC,1,1)
000814      000   C        FIND TOTAL NUMBER OF WIRES AND CONNECTED ELEMENTS
000815      000   C        TO THE ELEMENT AND SORT ELEMENTS IN ASCENDING ORDER
000816      000   C
000817      000        DO 107 IELT=1,NELT
000818      000        AUX1(IELT)=0
000819      000        AUX2(IELT)=0
000820      000        DO 107 JELT=1,NELT
000821      000        IF(ELTCON(IELT,JELT).EQ.0) GO TO 107
000822      000        AUX2(IELT)=AUX2(IELT)+1
000823      000        AUX1(IELT)=AUX1(IELT)+ELTCON(IELT,JELT)
```

```
000824      000    107   CONTINUE
000825      000          CALL SORT(AUX1,NWIRES,NELT,NELT,1,1)
000826      000          CALL SORT(AUX2,NCONS,NELT,NELT,1,1)
000827      000    C         SUM THE RANKS OF EACH ELEMENT (DESCENDING ORDER)
000828      000    C
000829      000          CALL SORT(NWIRES,AUX1,NELT,NELT,1,1)
000830      000          CALL SORT(NCONS,AUX2,NELT,NELT,1,1)
000831      000          DO 108 IELT=1,NELT
000832      000    108   AUX3(IELT)=AUX1(IELT)+AUX2(IELT)
000833      000          CALL SORT(AUX3,FINRAN,NELT,NELT,-1,1)
000834      000    C         FIND THE INITIAL LOCATION OF THE ELEMENTS
000835      000    C         ON THE ASSIGNMENT BOARD AND CALCULATE INITIAL BOUND
000836      000    C
000837      000          MELT=0
000838      000          DO 111 IELT=1,NELT
000839      000          ISWTCH=1
000840      000          IAS=FINRAN(IELT)
000841      000          IF(ASELT(IAS).EQ.EMPTY) GO TO 115
000842      000          GO TO 111
000843      000    C         FIND THE ELEMENT WHICH HAS MOST CONNECTIONS
000844      000    C         TO PREVIOUSLY ASSIGNED ELEMENT
000845      000    C
000846      000    113   ISWTCH=0
000847      000    C         ELEMENTS ARE SORTED IN DESCENDING ORDER ACCORDING
000848      000    C         TO MOST CONNECTIONS TO PREVIOUSLY ASSIGNED ELEMENTS
000849      000    C
000850      000          DO 114 J=1,NELT
000851      000    114   AUX1(J)=ELTCON(IAS,J)
000852      000          CALL SORT(AUX1,AUX2,NELT,NELT,-1,1)
000853      000          DO 118 I=1,NELT
000854      000          IF(AUX1(I).EQ.0) GO TO 111
000855      000          IAS=AUX2(I)
000856      000          IF(ASELT(IAS).EQ.EMPTY) GO TO 115
000857      000    118   CONTINUE
000858      000    C         MAKE THE ASSIGNMENTS TO NEXT BEST POSITION
000859      000    C         AND KEEP THE RECORD OF ASSIGNMENTS
000860      000    115   MELT=MELT+1
000861      000          ILO=IDISTS(MELT)
000862      000          BOARD(ILO)=IAS
000863      000          ASELT(IAS)=ILO
000864      000          ASRANK(MELT)=IAS
000865      000          NASSEL=NASSEL+1
000866      000          ASSEL(NASSEL)=IAS
000867      000          IF(MELT.EQ.NELT) GO TO 120
000868      000          IF(ISWTCH.EQ.0) GO TO 111
000869      000          GO TO 113
000870      000    111   CONTINUE
000871      000    120     DO 119 I=1,NLOC
000872      000    119   AUXBOA(I)=BOARD(I)
000873      000    C         SORT ELEMENT CONNECTIONS IN DESCENDING ORDER
000874      000    C
000875      000          NE=NELT-1
000876      000          KCON=0
000877      000          DO 116 I=1,NE
000878      000          IB=I+1
000879      000          DO 116 J=IB,NELT
000880      000          KCON=KCON+1
```

```
000881      000          IELTC3(KCON)=ELTCON(I,J)
000882      000          AUX5(KCON)=I
000883      000    116   AUX6(KCON)=J
000884      000          CALL SORT(IELTC3,AUX4,KCON,KCON,-1,1)
000885      000    C         MODIFY THE ELEMENT NUMBERS ACCORDING TO SORTING
000886      000    C
000887      000          DO 160 I=1,KCON
000888      000          IELT=AUX4(I)
000889      000          ELEFR(I)=AUX5(IELT)
000890      000    160   ELETO(I)=AUX6(IELT)
000891      000    C         SORT DISTANCES BETWEEN LOCATIONS IN ASCENDING ORDER
000892      000    C
000893      000          NE=NLOC-1
000894      000          KDIS=0
000895      000          DO 142 I=1,NE
000896      000          IB=I+1
000897      000          DO 142 J=IB,NLOC
000898      000          KDIS=KDIS+1
000899      000          ILOCC3(KDIS)=IDIST(I,J)
000900      000          AUX5(KDIS)=I
000901      000    142   AUX6(KDIS)=J
000902      000          CALL SORT(ILOCC3,AUX4,KDIS,KDIS,1,1)
000903      000    C         MODIFY THE LOCATION NUMBERS ACCORDING TO SORTING
000904      000    C
000905      000          DO 161 I=1,KDIS
000906      000          ILOC=AUX4(I)
000907      000          DISTFR(I)=AUX5(ILOC)
000908      000    161   DISTTO(I)=AUX6(ILOC)
000909      000    C         CALCULATE INITIAL CSTAR
000910      000    C
000911      000          DO 121 IELT=1,NELT
000912      000          ILOC=ASELT(IELT)
000913      000          K=IELT+1
000914      000          DO 121 JELT=K,NELT
000915      000          JLOC=ASELT(JELT)
000916      000    121   CSTAR=CSTAR+ELTCON(IELT,JELT)*IDIST(ILOC,JLOC)
000917      000          C1=CSTAR
000918      000          CPER=(IPERC*CSTAR)/100
000919      000          WRITE(IWT,1006) CSTAR
000920      000          DO 122 I=1,NLOC
000921      000    122   WRITE(IWT,1004) I,BOARD(I)
000922      000          ITER=NELT+1
000923      000    C         START ITERATIONS ON THE TREE
000924      000    C
000925      000    123   ITER=ITER-1
000926      000          IF(ITER.EQ.0) GO TO 200
000927      000    C         DELETE NEXT PREVIOUSLY ASSIGNED ELEMENT
000928      000    C
000929      000          IRE=ASRANK(ITER)
000930      000          ILO=ASELT(IRE)
000931      000          AUXBOA(ILO)=EMPTY
000932      000          ITREE(IRE,ILO)=INFIN
000933      000          ASELT(IRE)=EMPTY
000934      000          NASSEL=NASSEL-1
000935      000    C         CALCULATE COST OF DELETED ELEMENT
000936      000    C
000937      000    C         UPDATE C1
```

```
000935      000    C
000939      000          DO 140 I=1,NASSEL
000940      000          IELT=ASSEL(I)
000941      000          ILOC=ASELT(IELT)
000942      000    140   C1=C1-ELTCON(IRE,IELT)*IDIST(ILO,ILOC)
000943      000    C          CALCULATE C2 & C3
000944      000    C
000945      000          C2=0
000946      000          C3=0
000947      000          IF(NASSEL.LT.NELT) CALL COST(C2,C3,IC2)
000948      000          CNEW=C1+C2+C3
000949      000          I10=I10+1
000950      000          IF(I10.LE.10) WRITE(IWT,1007) I10,C1,C2,C3
000951      000          IF(CNEW.LT.CPER) GO TO 126
000952      000    124   DO 125 I=1,NLOC
000953      000    125   ITREE(IRE,I)=EMPTY
000954      000          GO TO 123
000955      000    C          CHECK IF THERE IS A POSSIBILITY TO ASSIGN THE
000956      000    C          REMOVED ELEMENT TO ANY OTHER LOCATION
000957      000    126   DO 130 JLOC=1,NLOC
000958      000          IF(AUXBOA(JLOC).EQ.EMPTY.AND.ITREE(IRE,JLOC).EQ.EMPTY)
000959      000         1 GO TO 127
000960      000          GO TO 130
000961      000    C          MAKE THE NEW ASSIGNMENT
000962      000    C
000963      000    127   AUXBOA(JLOC)=IRE
000964      000          ASELT(IRE)=JLOC
000965      000          ITER=ITER+1
000966      000          NASSEL=NASSEL+1
000967      000    C          CALCULATE COST OF NEW ASSIGNED ELEMENT
000968      000    C
000969      000    C          UPDATE C1
000970      000    C
000971      000          DO 150 I=1,NASSEL
000972      000          IELT=ASSEL(I)
000973      000          ILOC=ASELT(IELT)
000974      000    150   C1=C1+ELTCON(IRE,IELT)*IDIST(JLOC,ILOC)
000975      000          C2=0
000976      000          C3=0
000977      000          IF(NASSEL.LT.NELT) CALL COST(C2,C3,IC2)
000978      000          CNEW=C1+C2+C3
000979      000          I10=I10+1
000980      000          IF(I10.LE.10) WRITE(IWT,1007) I10,C1,C2,C3
000981      000          IF(CNEW.GE.CPER) GO TO 123
000982      000    C          CHECK IF ALL THE ELEMENTS HAVE BEEN ASSIGNED
000983      000    C
000984      000          IF(ITER.EQ.NELT+1) GO TO 132
000985      000          GO TO 131
000986      000    130   CONTINUE
000987      000          GO TO 124
000988      000    C          FIND THE NEXT UNASSIGNED ELEMENT
000989      000    C
000990      000    131   IRE=ASRANK(ITER)
000991      000          GO TO 126
000992      000    C          UPDATE ASSIGNMENT BOARD & BOUND
000993      000    C
000994      000    132   CSTAR=CNEW
```

```
000995      000          CPER=(IPERC*CSTAR)/100
000996      000          DO 133 IJ=1,NLOC
000997      000    133   BOARD(IJ)=AUXBOA(IJ)
000998      000          GO TO 123
000999      000    C         OUTPUT THE RESULTS
001000      000    C
001001      000    200   WRITE(IWT,1005) CSTAR
001002      000          DO 134 I=1,NLOC
001003      000    134   WRITE(IWT,1004) I,BOARD(I)
001004      000    C         FORMAT STATEMENTS
001005      000    C
001006      000    1001  FORMAT(8I5)
001007      000    1002  FORMAT(I5,5(I5,I10))
001008      000    1003  FORMAT(10X,7I10)
001009      000    1004  FORMAT(2X,I3,9X,I3)
001010      000    1005  FORMAT(///,2X,'OPTIMAL COST=',I8,///1X,'LOCATION',4X,'ELEMENT')
001011      000    1006  FORMAT(//,2X,'INITIAL COST=',I8,///1X,'LOCATION',4X,'ELEMENT')
001012      000    1007  FORMAT(/2X,'ITERATION:',I3,'  C1:',I5,'  C2:',I5,'  C3:',I5)
001013      000    9999  FORMAT( )
001014      000          STOP
001015      000          END
001016      000    @ELT,SI TUMER-S*SINAN-BAB,COST...17061504131Z
001017      000          SUBROUTINE COST(C2,C3,IC2)
001018      000    C
001019      000    C         ROUTINE CALCULATES THE COST OF THE QUADRATIC BOUND ASSIGNMENT
001020      000    C         PROBLEM AND ALSO CALCULATES THE COST OF INITIAL SOLUTION
001021      000    C
001022      000    C         VARIABLES:
001023      000    C         ASSEL :ASSIGNED ELEMENTS
001024      000    C         C2    :COST OF ASSIGNMENTS TO UNASSIGNED ONES
001025      000    C         C3    :COST OF ASSIGNMENTS TO EACHOTHER
001026      000    C         IC2   :=1 IF C2 IS CALCULATED WITH RHO MATRIX
001027      000    C                =0 IF C2 IS CALCULATED BY SORTED ELEMENTS AND LOCATIONS
001028      000    C         NASSEL:NUMBER OF ASSIGNED ELEMENTS
001029      000    C         NUNASE:NUMBER OF UNASSIGNED ELEMENTS
001030      000    C         NUNASL:NUMBER OF UNASSIGNED LOCATIONS
001031      000    C         RHO   :RHO MATRIX FOR CALCULATING OF C2
001032      000    C         UNASEL:UNASSIGNED ELEMENTS
001033      000    C         UNASLO:LOCATIONS WHICH ARE UNASSIGNED
001034      000    C
001035      000    C
001036      000          COMMON/AUXILI/AUX5(800),AUX6(800)
001037      000          COMMON/AREA/AUXBOA(50),AUX1(50),AUX2(40),AUX3(40),ITREE(35,40),
001038      000         1IDIST(40,40),ELTCON(35,40),ASSELT(35),ASSEL(35),NELT,NLOC,NASSEL
001039      000         2,KDIS,KCON,IELTC3(650),IELTC2(800),ELEFR(650),ELETO(650),
001040      000         3DISTFR(800),DISTTO(800)
001041      000          INTEGER AUXBOA,AUX1,AUX2,AUX3,EMPTY,ELTCON,RHO(35,40),UNASEL(35),
001042      000         1UNASLO(40),ASSEL,ASSELT,AUX5,AUX6,C2,C3
001043      000         2,ELEFR,ELETO,DISTFR,DISTTO,FROM,TO
001044      000    C
001045      000    C         FIND ASSIGNED AND UNASSIGNED ELEMENTS & LOCATIONS
001046      000    C
001047      000          IFIN=1
001048      000          IWT=6
001049      000    9999  FORMAT( )
001050      000          EMPTY=0
001051      000          IF(IC2.EQ.0) GO TO 112
```

```
001052      000          NUNASE=0
001053      000          NUNASL=0
001054      000          DO 90 I=1,NELT
001055      000          IF(ASELT(I).EQ.EMPTY) GO TO 91
001056      000          GO TO 90
001057      000    C         FIND UNASSIGNED ELEMENTS
001058      000    C
001059      000    91    NUNASE=NUNASE+1
001060      000          UNASEL(NUNASE)=I
001061      000    90    CONTINUE
001062      000    C         FIND UNASSIGNED LOCATIONS
001063      000    C
001064      000          DO 92 I=1,NLOC
001065      000          IF(AUXBOA(I).NE.EMPTY) GO TO 92
001066      000          NUNASL=NUNASL+1
001067      000          UNASLO(NUNASL)=I
001068      000    92    CONTINUE
001069      000          IF(NASSEL.EQ.0) GO TO 110
001070      000    C         CALCULATE C2
001071      000    C
001072      000    C         CREATE RHO MATRIX
001073      000    C
001074      000    101   DO 102 I=1,NUNASE
001075      000          M=UNASEL(I)
001076      000          DO 102 J=1,NUNASL
001077      000          L=UNASLO(J)
001078      000    C         CHECK IF WE ARE ALLOWED TO ASSIGN ELEMENT M TO LOCATION L
001079      000    C
001080      000          IF(ITREE(M,L).EQ.EMPTY) GO TO 99
001081      000          RHO(I,J)=ITREE(M,L)
001082      000          GO TO 102
001083      000    99    RHO(I,J)=0
001084      000          DO 98 K=1,NASSEL
001085      000          IE=ASSEL(K)
001086      000          IL=ASELT(IE)
001087      000          RHO(I,J)=RHO(I,J)+ELTCON(M,IE)*IDIST(L,IL)
001088      000    98    CONTINUE
001089      000    102   CONTINUE
001090      000    C         FIND MINIMUMS OF ROWS (STORED IN AUX1)
001091      000    C
001092      000          DO 103 I=1,NUNASE
001093      000          AUX1(I)=RHO(I,1)
001094      000          DO 103 J=2,NUNASL
001095      000          IF(AUX1(I).LE.RHO(I,J)) GO TO103
001096      000          AUX1(I)=RHO(I,J)
001097      000    103   CONTINUE
001098      000          DO 105 I=1,NUNASE
001099      000          DO 105 J=1,NUNASL
001100      000    105   RHO(I,J)=RHO(I,J)-AUX1(I)
001101      000    C         FIND MINIMUMS OF COLUMNS (STORED IN AUX2)
001102      000    C
001103      000          DO 104 J=1,NUNASL
001104      000          AUX2(J)=RHO(1,J)
001105      000          DO 104 I=2,NUNASE
001106      000          IF(AUX2(J).LE.RHO(I,J)) GO TO 104
001107      000          AUX2(J)=RHO(I,J)
001108      000    104   CONTINUE
```

```
001109      000   C          ADD MINIMUMS OF ROWS AND COLUMNS
001110      000   C
001111      000         DO 106 I=1,NUNASE
001112      000   106   C2=C2+AUX1(I)+AUX2(I)
001113      000         IF(NUNASE.LE.1) RETURN
001114      000         GO TO 110
001115      000   C          CALCULATE C2 BY SORTING METHOD
001116      000   C
001117      000   112   K=0
001118      000         IF(NASSEL.EQ.0) GO TO 110
001119      000         DO 116 I=1,KCON
001120      000   C          CHECK WHETHER EITHER ONE OF THE ELEMENTS ASSIGNED
001121      000   C
001122      000         FROM=ELEFR(I)
001123      000         TO=ELETO(I)
001124      000         IF(ASELT(FROM).EQ.EMPTY) GO TO 113
001125      000         IF(ASELT(TO).NE.EMPTY) GO TO 116
001126      000         GO TO 114
001127      000   113   IF(ASELT(TO).EQ.EMPTY) GO TO 116
001128      000         K=K+1
001129      000         AUX6(K)=FROM
001130      000         GO TO 115
001131      000   114   K=K+1
001132      000         AUX6(K)=TO
001133      000   115   AUX5(K)=IELTC3(I)
001134      000   116   CONTINUE
001135      000   C          CHECK WHETHER EITHER ONE OF THE LOCATIONS ASSIGNED
001136      000   C
001137      000         M=0
001138      000         DO 130 I=1,KDIS
001139      000         FROM=DISTFR(I)
001140      000         TO=DISTTO(I)
001141      000         IF(AUXBOA(FROM).EQ.EMPTY) GO TO 121
001142      000         IF(AUXBOA(TO).NE.EMPTY) GO TO 130
001143      000         GO TO 123
001144      000   121   IF(ASELT(TO).EQ.EMPTY) GO TO 130
001145      000         M=M+1
001146      000         IT=FROM
001147      000         GO TO 124
001148      000   123   M=M+1
001149      000         IT=TO
001150      000   124   IJ=AUX6(M)
001151      000         IF(ITREE(IJ,IT).NE.EMPTY) GO TO 130
001152      000         C2=C2+AUX5(M)*ILOCC3(I)
001153      000         IF(M.EQ.K) GO TO 110
001154      000   130   CONTINUE
001155      000   C          CALCULATE C3
001156      000   C
001157      000   110   K=0
001158      000         DO 135 I=1,KCON
001159      000   C          CHECK WHETHER BOTH THE ELEMENTS ARE ASSIGNED
001160      000   C
001161      000         FROM=ELEFR(I)
001162      000         TO=ELETO(I)
001163      000         IF(ASELT(FROM).EQ.EMPTY.AND.ASELT(TO).EQ.EMPTY) GO TO 136
001164      000         GO TO 135
001165      000   136   K=K+1
```

```
001160  000        AUX5(K)=IELTC3(I)
001167  000  135   CONTINUE
001168  000  C         CHECK WHETHER BOTH THE LOCATIONS ARE UNASSIGNED
001169  000  C
001170  000        M=0
001171  000        DO 139 I=1,KDIS
001172  000        FROM=DISTFR(I)
001173  000        TO=DISTTO(I)
001174  000        IF(AUXBOA(FROM).EQ.EMPTY.AND.AUXBOA(TO).EQ.EMPTY) GO TO 141
001175  000        GO TO 139
001176  000  141   M=M+1
001177  000        C3=C3+AUX5(M)*ILOCC3(I)
001178  000        IF(M.EQ.K) GO TO 150
001179  000  139   CONTINUE
001180  000  150   RETURN
001181  000        END
001182  000        SUBROUTINE SORT(WORK,IWORK,NELT,KELT,KOD1,INXSRT)
001183  000  C
001184  000  C         SORTING ROUTINE OF AN INTEGER VECTOR
001185  000  C         PROGRAM SORTS THE ELEMENTS OF A VECTOR AND INDEXES
001186  000  C         ACCORDING TO ASCENDING OR DESCENDING ORDER
001187  000  C
001188  000  C         VARIABLES:
001189  000  C         WORK  :VECTOR TO BE SORTED
001190  000  C         IWORK :SORTED INDEXES
001191  000  C         WAIT  :TEMPORARY AREA
001192  000  C         KELT  :NUMBER OF ELEMENTS IN THE VECTOR IWORK
001193  000  C         NELT  :NUMBER OF ELEMENTS IN THE VECTOR WORK
001194  000  C         KOD1  :=1 , VECTOR IS TO BE SORTED IN ASCENDING ORDER
001195  000  C               :=-1 , VECTOR IS TO BE SORTED IN DESCENDING ORDER
001196  000  C         SMALL :SMALLEST ELEMENT IN THE VECTOR AT THE MOMENT OF SORTING
001197  000  C         I     :ITERATION NUMBER
001198  000  C         INXSRT:=1 , INDEX SORTING IS DESIRED
001199  000  C                =0 OTHERWISE
001200  000  C         J     :ELEMENT NUMBER
001201  000  C
001202  000  C
001203  000        INTEGER WORK,WAIT,SMALL
001204  000        DIMENSION WORK(NELT),IWORK(KELT)
001205  000        IWT=6
001206  000  C         WHEN INDEX SORTING IS DESIRED NELT MUST BE EQUAL TO KELT
001207  000  C
001208  000        IF(INXSRT.EQ.0) GO TO 96
001209  000        IF(NELT.EQ.KELT) GO TO 94
001210  000        WRITE(IWT,1002) INXSRT
001211  000  1002  FORMAT(//' STOP: CHECK THE DIMENSIONS OF ARRAYS TO BE SORTED'
001212  000       1,//'  INXSRT=',I6)
001213  000        STOP
001214  000  C         INITIALIZE INDEX ARRAY
001215  000  C
001216  000  94    DO 95 IELT=1,NELT
001217  000  95    IWORK(IELT)=IELT
001218  000  96    I=1
001219  000        IF(INXSRT.EQ.0) INS=1
001220  000        IF(INXSRT.EQ.1) INS=2
001221  000        NELT1=NELT+1
001222  000  100   J=I
```

```
001225      000   C          INITIALIZE THE SMALLEST VALUE OF NEXT ITERATION ARBITRARILY
001224      000   C
001225      000         SMALL=WORK(I)
001226      000   102   J=J+1
001227      000         IF(J.EQ.NELT1) GO TO 101
001228      000   C          CHECK WHETHER NEXT ELEMENT IS LESS THAN THE
001229      000   C          PRESENT SMALL VALUE
001230      000   C
001231      000         IF(SMALL.LE.WORK(J)) GO TO 102
001232      000   C          CHANGE THE ELEMENTS ACCORDING TO ASCENDING ORDER
001233      000   C
001234      000         SMALL=WORK(J)
001235      000         WAIT=WORK(I)
001236      000         WORK(I)=WORK(J)
001237      000         WORK(J)=WAIT
001238      000   C          CHANGE INDEXES TO ASCENDING ORDER
001239      000   C
001240      000         GO TO (102,103),INS
001241      000   103   WAIT=IWORK(I)
001242      000         IWORK(I)=IWORK(J)
001243      000         IWORK(J)=WAIT
001244      000         GO TO 102
001245      000   C          NEXT ITERATION
001246      000   C
001247      000   101   I=I+1
001248      000   C          END OF ITERATION
001249      000   C
001250      000         IF(I.LT.NELT1) GO TO 100
001251      000   C          CHECK IF DESCENDING ORDER IS DESIRED
001252      000   C
001253      000         IF(KOD1.EQ.1) RETURN
001254      000         IF(KOD1.EQ.-1) GO TO 106
001255      000         WRITE(IWT,1001)
001256      000   1001  FORMAT(//,5X,'STOP KOD1 IS NOT PROPER')
001257      000         STOP
001258      000   C          REARRANGE THE ELEMENTS AND INDEXES IN DESCENDING ORDER
001259      000   C
001260      000   106   NB=1
001261      000         NE=NELT
001262      000         N=NELT/2
001263      000         DO 107 M=1,N
001264      000         WAIT=WORK(NE)
001265      000         WORK(NE)=WORK(NB)
001266      000         WORK(NB)=WAIT
001267      000         GO TO (105,104),INS
001268      000   104   WAIT=IWORK(NE)
001269      000         IWORK(NE)=IWORK(NB)
001270      000         IWORK(NB)=WAIT
001271      000   105   NB=NB+1
001272      000   107   NE=NE-1
001273      000         RETURN
001274      000         END
001275      000   @FOR,IS MAIN
001276      000   @ELT,SI TUMER-S*SINAN-QAP5,MAIN,,,143216053312
001277      000         COMMON IC1,IC2,IC3,NFCIL,NLOCT,N2,M2,LPCST,ITOT                    QAP3   1
001278      000         COMMON MSTER(04000),IWORK(2000),KSNSW(60)                          QAP3   2
001279      000         COMMON KLBND,KUPDT                                                 QAP3   3
```

```
001280  000        INTEGER F(20,20),D(20,20),DISTI(400),DISTJ(400),INTRI(400)                QAP3   4
001281  000        INTEGER INTRJ(400),LEVEL(20),LOCAT(20),FCLTY(20),PTCST(20)                QAP3   5
001282  000        INTEGER FIXED(20),WORK1(20,20),CMCST(20),LIKE(20)                         QAP3   6
001283  000        INTEGER IPASS(20),ICOST(100),ILOCT(100,20),FOUND(100)                     QAP3   7
001284  000        INTEGER   FIX(20),WHERE(20)                                               QAP3   8
001285  000        LOGICAL FCFLG(20),FLOST(20),FLGDS(400),FLGIN(400),BANED(20,20)            QAP3   9
001286  000   C                                                                              QAP3  10
001287  000        LOGICAL WORK2(1300),WORK8(1300)                                           QAP3  11
001288  000        INTEGER WORK3(0120),WORK7(0120),WORK6(0020)                               QAP3  12
001289  000   C                                                                              QAP3  13
001290  000   C                                                                              QAP3  14
001291  000        EQUIVALENCE( MSTER(0001),F(1,1))                                          QAP3  15
001292  000        EQUIVALENCE( MSTER(0401),D(1,1))                                          QAP3  16
001293  000        EQUIVALENCE( MSTER(0801),DISTI(1))                                        QAP3  17
001294  000        EQUIVALENCE( MSTER(1201),DISTJ(1))                                        QAP3  18
001295  000        EQUIVALENCE( MSTER(1601),INTRI(1))                                        QAP3  19
001296  000        EQUIVALENCE( MSTER(2001),INTRJ(1))                                        QAP3  20
001297  000        EQUIVALENCE( MSTER(2401),LEVEL(1))                                        QAP3  21
001298  000        EQUIVALENCE( MSTER(2421),LOCAT(1))                                        QAP3  22
001299  000        EQUIVALENCE( MSTER(2441),FCLTY(1))                                        QAP3  23
001300  000        EQUIVALENCE( MSTER(2461),PTCST(1))                                        QAP3  24
001301  000        EQUIVALENCE( MSTER(2481),FCFLG(1))                                        QAP3  25
001302  000        EQUIVALENCE( MSTER(2501),FLOST(1))                                        QAP3  26
001303  000        EQUIVALENCE( MSTER(2521),FLGDS(1))                                        QAP3  27
001304  000        EQUIVALENCE( MSTER(2921),FLGIN(1))                                        QAP3  28
001305  000        EQUIVALENCE( MSTER(2941),FIXED(1))                                        QAP3  29
001306  000        EQUIVALENCE( MSTER(3001),CMCST(1))                                        QAP3  30
001307  000        EQUIVALENCE( MSTER(3401),WORK1(1,1))                                      QAP3  31
001308  000        EQUIVALENCE( MSTER(3821),LIKE(1))                                         QAP3  32
001309  000        EQUIVALENCE( MSTER(3841),  FIX(1))                                        QAP3  33
001310  000   C                                                                              QAP3  34
001311  000        EQUIVALENCE( IWORK(0001),WORK1(1,1))                                      QAP3  35
001312  000   C                                                                              QAP3  36
001313  000        EQUIVALENCE( MSTER(2401),WORK7(1))                                        QAP3  37
001314  000        EQUIVALENCE( MSTER(2521),WORK8(1))                                        QAP3  38
001315  000   C                                                                              QAP3  39
001316  000   C                                                                              QAP3  40
001317  000   C                                                                              QAP3  41
001318  000   C    READ THE KEYS SETTINGS FOR THE MAIN PROGRAM                               QAP3  42
001319  000          READ(5,3)(KSNSW(I),I=01,20)                                           QAP3  43
001320  000   C    READ THE KEY SETTINGS FOR SUBROUTINE LWBND                                QAP3  44
001321  000          READ(5,3)(KSNSW(I),I=21,30)                                           QAP3  45
001322  000   C    READ THE PROBLEM DATA                                                     QAP3  46
001323  000       10 CALL DTAIN(JOB,IFRAC,NFIXD,ISTOP,MORE,ITERM)                          QAP3  47
001324  000   C                                                                              QAP3  48
001325  000   C    PRINT THE INITIAL DATA IF YOU WISH                                        QAP3  49
001326  000          IF(KSNSW(01).EQ.0)GO TO 110                                           QAP3  50
001327  000          WRITE(6,4)JOB,NFCIL,NLOCT                                             QAP3  51
001328  000          DO 70 I=1,2                                                           QAP3  52
001329  000       70 WRITE(6,2)                                                            QAP3  53
001330  000          WRITE(6,7)                                                            QAP3  54
001331  000          WRITE(6,5)                                                            QAP3  55
001332  000          DO 80 I=1,NLOCT                                                       QAP3  56
001333  000       80 WRITE(6,01)(D(I,J),J=1,NLOCT)                                         QAP3  57
001334  000          WRITE(6,8)                                                            QAP3  58
001335  000          DO 90 I=1,NFCIL                                                       QAP3  59
001336  000       90 WRITE(6,01)(F(I,J),J=1,NFCIL)                                         QAP3  60
```

```
001337  000          WRITE(6,7)                                                  QAP3  61
001338  000          WRITE(6,51)LBCST,ITOT                                       QAP3  62
001339  000    C   INITIALIZE THE COST                                           QAP3  63
001340  000      110 IC1=0                                                       QAP3  64
001341  000          N2=NLOCT*NLOCT                                              QAP3  65
001342  000          M2=NFCIL*NFCIL                                              QAP3  66
001343  000          IF(NFIXD.EQ.1)GO TO 125                                     QAP3  67
001344  000          DO 120 I=1,10                                               QAP3  68
001345  000      120 WRITE(6,98)NFIXD                                            QAP3  69
001346  000          GO TO 900                                                   QAP3  70
001347  000    C   UNFLAG ALL THE LOCATIONS AND THE FACILITIES                   QAP3  71
001348  000      125 DO 130 I=1,NLOCT                                            QAP3  72
001349  000      130 FLOGT(I)=.FALSE.                                            QAP3  73
001350  000          DO 140 J=1,NFCIL                                            QAP3  74
001351  000      140 FCFLG(J)=.FALSE.                                            QAP3  75
001352  000    C                                                                 QAP3  76
001353  000          DO 150 I=1,NFCIL                                            QAP3  77
001354  000          IPASS(I)=0                                                  QAP3  78
001355  000          DO 150 J=1,NLOCT                                            QAP3  79
001356  000      150 DANLD(I,J)=.FALSE.                                          QAP3  80
001357  000    C   RANK THE DISTANCES IN AN ASCENDING ORDER                      QAP3  81
001358  000          CALL ARRNG(N2,1,2)                                          QAP3  82
001359  000    C   PRINT THE MATRIX OF ARRANGED DISTANCES                        QAP3  83
001360  000          IF(KSNSW(02).EQ.0)GO TO 200                                 QAP3  84
001361  000          WRITE(6,21)                                                 QAP3  85
001362  000          WRITE(6,01)(DISTI(I),I=1,M2)                                QAP3  86
001363  000          WRITE(6,07)                                                 QAP3  87
001364  000          WRITE(6,01)(DISTJ(I),I=1,M2)                                QAP3  88
001365  000    C                                                                 QAP3  89
001366  000    C   FLAG ALL THE DISTANCES                                        QAP3  90
001367  000      200 DO 210 I=1,N2                                               QAP3  91
001368  000      210 FLGDS(I)=.FALSE.                                            QAP3  92
001369  000    C                                                                 QAP3  93
001370  000    C   RANK THE INTERACTIONS IN A DESCENDING ORDER                   QAP3  94
001371  000          CALL ARRNG(M2,2,2)                                          QAP3  95
001372  000    C   PRINT THE INTERACTIONS IF YOU WISH                            QAP3  96
001373  000          IF(KSNSW(03).EQ.0)GO TO 310                                 QAP3  97
001374  000          WRITE(6,22)                                                 QAP3  98
001375  000          WRITE(6,01)(INTRI(I),I=1,M2)                                QAP3  99
001376  000          WRITE(6,07)                                                 QAP3 100
001377  000          WRITE(6,01)(INTRJ(I),I=1,M2)                                QAP3 101
001378  000    C                                                                 QAP3 102
001379  000    C   FLAGG ALL THE INTERACTIONS                                    QAP3 103
001380  000      310 DO 320 I=1,M2                                               QAP3 104
001381  000      320 FLGIN(I)=.FALSE.                                            QAP3 105
001382  000    C   INITIALIZE WORK7                                              QAP3 106
001383  000          DO 325 I=1,120                                              QAP3 107
001384  000      325 WORK7(I)=0                                                  QAP3 108
001385  000    C   INITIALIZE WORK8                                              QAP3 109
001386  000          DO 330 I=1,1300                                             QAP3 110
001387  000      330 WORK8(I)=.FALSE.                                            QAP3 111
001388  000          DO 335 I=1,20                                               QAP3 112
001389  000      335 CMCST(I)=0                                                  QAP3 113
001390  000    C                                                                 QAP3 114
001391  000    C                                                                 QAP3 115
001392  000          TIME=0                                                      QAP3 116
001393  000          IEVAL=0                                                     QAP3 117
```

```
001394  000        IUBCS=0                                                    QAP3 118
001395  000        KOMPL=0                                                    QAP3 119
001396  000        FRAC=IFRAC                                                 QAP3 120
001397  000        FRAC=0.01*FRAC                                             QAP3 121
001398  000  C                                                                QAP3 122
001399  000  C   INITIALIZE COUNTERS                                          QAP3 123
001400  000        KLBND=0                                                    QAP3 124
001401  000        KUPDT=0                                                    QAP3 125
001402  000        KPATH=0                                                    QAP3 126
001403  000        KTERM=0
001404  000        KOMIT=100*NFCIL
001405  000  C                                                                QAP3 127
001406  000  C   CALCULATE A STARTING LOWER BOUND                             QAP3 128
001407  000        IF(LBCST.GT.0)GO TO 340                                    QAP3 129
001408  000        CALL LWBND(0,0,0,0)                                        QAP3 130
001409  000      340 LBTC=LBCST                                               QAP3 131
001410  000        IF(KSNSW(04).EQ.1)WRITE(6,17)LBTC                          QAP3 132
001411  000  C                                                                QAP3 133
001412  000  C     PROCEED TO OBTAIN A FIRST FEASIBLE SOLUTION ALONG THE RIGHTMOST  QAP3 134
001413  000  C   BRANCH OR THE TREE                                           QAP3 135
001414  000  C   A) MAKE THW ASSIGNMENT OF THE FIRST ELEMENT TO THE FIRST LOCATION  QAP3 136
001415  000        ITER=1                                                     QAP3 137
001416  000        LEVY=1                                                     QAP3 138
001417  000        L2=  FIX(1)                                                QAP3 139
001418  000         J=WHERE(1)                                                QAP3 140
001419  000        LEVEL(L2)=1                                                QAP3 141
001420  000        FIXED(1)=L2                                                QAP3 142
001421  000        LOCAT(L2)=J                                                QAP3 143
001422  000        LIXED(1)=J                                                 QAP3 144
001423  000        PTCST(1)=0                                                 QAP3 145
001424  000        CMCST(1)=0                                                 QAP3 146
001425  000        FCLTY(1)=L2                                                QAP3 147
001426  000        FCFLG(L2)=.TRUE.                                           QAP3 148
001427  000        FLOGT(J)=.TRUE.                                            QAP3 149
001428  000        L1=J                                                       QAP3 150
001429  000        IF(KSNSW(05).EQ.1)WRITE(6,9)ITER,LEVY,L2,L1                QAP3 151
001430  000        CALL UPDATE(L2,L1,2)                                       QAP3 152
001431  000  C   B) CHOOSE THE ELEMENT WITH MAXIMUM INTERACTIONS WITH THE ONE ASSIGNEQAP3 153
001432  000
001433  000
001434  000  C   AT THE PRECEDING LEVEL                                       QAP3 154
001435  000  + 43? L5VY=L5VY+A++ +  + +                                      QAP3 155
001436  000        IPASS(LEVY)=IPASS(LEVY)+1                                  QAP3 156
001437  000
001438  000        ITER=ITER+1                                                QAP3 157
001439  000        JET=FCLTY(LEVY-1)                                          QAP3 158
001440  000        ISAV=-32767                                                QAP3 159
001441  000        DO 460 I=1,NFCIL                                           QAP3 160
001442  000        IF(FCFLG(I))GO TO 460                                      QAP3 161
001443  000        IF(F(I,JET).LE.ISAV.AND.F(JET,I).LE.ISAV)GO TO 460         QAP3 162
001444  000        ISAV=F(I,JET)                                              QAP3 163
001445  000        IF(F(JET,I).GT.F(I,JET))ISAV=F(JET,I)                      QAP3 164
001446  000        KEEP=I                                                     QAP3 165
001447  000      460 CONTINUE                                                 QAP3 166
001448  000      470 FCFLG(KEEP)=.TRUE.                                       QAP3 167
001449  000        LEVEL(KEEP)=LEVY                                           QAP3 168
001450  000  C   C) ASSIGN THE ELEMENT TO A LOCATION SO THAT THE COST OF SUCH AN ASSIQAP3 169
```

```
001451  000   C       IS MINIMAL                                                  GAP3 170
001452  000     510 IWD=32767                                                   GAP3 171
001453  000         DO 530 J=1,NLOCT                                            GAP3 172
001454  000         IF(DANED(KEEP,J))GO TO 530                                  GAP3 173
001455  000         IF(FLOGT(J))GO TO 530                                       GAP3 174
001456  000         KAM=0                                                       GAP3 175
001457  000         DO 520 I=1,NFCIL                                            GAP3 176
001458  000         IF(I.EQ.KEEP)GO TO 520                                      GAP3 177
001459  000         IF(.NOT.FCFLG(I))GO TO 520                                  GAP3 178
001460  000         L=LOCAT(I)                                                  GAP3 179
001461  000         ISAVV=F(KEEP,I)                                             GAP3 180
001462  000         IF(F(I,KEEP).GT.F(KEEP,I))ISAVV=F(I,KEEP)                   GAP3 181
001463  000         KAM=KAM+(ISAVV*D(J,L))                                      GAP3 182
001464  000     520 CONTINUE                                                    GAP3 183
001465  000         IF(KAM.GT.IWD)GO TO 530                                     GAP3 184
001466  000         IWD=KAM                                                     GAP3 185
001467  000         LOCAT(KEEP)=J                                               GAP3 186
001468  000     530 CONTINUE                                                    GAP3 187
001469  000         IF(IWD.EQ.32767)GO TO 700                                   GAP3 188
001470  000         IF(KSNSW(05).EQ.1)WRITE(6,9)ITER,LEVY,KEEP,LOCAT(KEEP),IWD  GAP3 189
001471  000         J=LOCAT(KEEP)                                               GAP3 190
001472  000         FCLTY(LEVY)=KEEP                                            GAP3 191
001473  000         FLOGT(J)=.TRUE.                                             GAP3 192
001474  000         IUBCS=IUBCS+IWD                                             GAP3 193
001475  000         FIXED(LEVY)=KEEP                                            GAP3 194
001476  000         PTCST(LEVY)=IWD                                             GAP3 195
001477  000         CALL UPDATE(KEEP,J,2)                                       GAP3 196
001478  000         IF(KSNSW(06).EQ.0)GO TO 580                                 GAP3 197
001479  000         CMCST(LEVY)=IUBCS                                           GAP3 198
001480  000         LIXED(LEVY)=J                                               GAP3 199
001481  000   C                                                                 GAP3 200
001482  000   C   CHECK WHETHER WE HAVE REACHED A TERMINAL NODE OR NOT         GAP3 201
001483  000     560 IF( LEVY.LT.NFCIL.AND.IWD.NE.0)GO TO 430                    GAP3 202
001484  000         IF( LEVY.EQ.NFCIL)GO TO 590                                 GAP3 203
001485  000   C   CHECK WHETHER IT PAYS TO CONTINUE THE SEARCH ALONG THIS BRANCH GAP3 204
001486  000         CALL LWBND(LEVY,ITOT-IUBCS-1)                               GAP3 205
001487  000         IF(LBCST.LT.ITOT)GO TO 565                                  GAP3 206
001488  000         KPATH=KPATH+1                                               GAP3 207
001489  000         GO TO 620                                                   GAP3 208
001490  000     565 LEVY=LEVY+1                                                 GAP3 209
001491  000         IPASS(LEVY)=IPASS(LEVY)+1                                   GAP3 210
001492  000         GO TO 730                                                   GAP3 211
001493  000   C   E) NOW WE HAVE A COMPLETE SOLUTION , PRINT IT IF YOU WISH    GAP3 212
001494  000     590 KOMPL=1                                                     GAP3 213
001495  000         IC1=IUBCS                                                   GAP3 214
001496  000         IF(IUBCS.GE.ITOT)GO TO 620                                  GAP3 215
001497  000         ITOT=IUBCS                                                  GAP3 216
001498  000         IEVAL=IEVAL+1                                               GAP3 217
001499  000         ICOST(IEVAL)=IUBCS                                          GAP3 218
001500  000         FOUND(IEVAL)=ITER                                           GAP3 219
001501  000         DO 595 I=1,NFCIL                                            GAP3 220
001502  000         ILOCT(IEVAL,I)=LOCAT(I)                                     GAP3 221
001503  000     595 CONTINUE                                                    GAP3 222
001504  000     600 IF(KSNSW(06).EQ.0)GO TO 610                                 GAP3 223
001505  000         WRITE(6,11)ITER                                             GAP3 224
001506  000         WRITE(6,12)                                                 GAP3 225
001507  000         DO 605 I=1,LEVY                                             GAP3 226
```

```
001508  000      605 WRITE(6,14)I,FIXED(I),LIXED(I),PTCST(I),CMCST(I)                QAP3 227
001509  000          CALL TIMOUT
001510  000    C                                                                     QAP3 228
001511  000    C                                                                     QAP3 229
001512  000    C                                                                     QAP3 230
001513  000    C   REDUCE THE CURRENT LOWER BOUND                                   QAP3 231
001514  000      610 DIF=ITOT-LBTC                                                   QAP3 232
001515  000          KETOT=ITOT                                                      QAP3 233
001516  000          IF(ITOT.LE.LBTC)GO TO 800
001517  000          KELBC=LBTC                                                      QAP3 234
001518  000          IF(DIF.LE.ISTOP)KTERM=100
001519  000          IF(DIF.LE.ISTOP)GO TO 611                                       QAP3 235
001520  000          DIF=(FRAC*DIF)                                                  QAP3 236
001521  000          ITOT=LBTC+DIF                                                   QAP3 237
001522  000      611 IF(KSNSW(08).EQ.1)WRITE(6,36)KELBC,ITOT,KETOT                   QAP3 238
001523  000    C                                                                     QAP3 239
001524  000    C   F)STORE THE CONFIGURATION OF THE CURRENT NODE SO THAT WE CAN RESTORQAP3 240
001525  000    C     IT IF WE FAIL TO OBTAIN A BETTER SOLUTION UNDER THE CURRENT CEILINQAP3 241
001526  000          DO 612 I=1,1300                                                 QAP3 242
001527  000          WORK2(I)=.FALSE.                                                QAP3 243
001528  000          IF(WORK8(I))WORK2(I)=.TRUE.                                     QAP3 244
001529  000      612 CONTINUE                                                        QAP3 245
001530  000          DO 615 I=1,120                                                  QAP3 246
001531  000      615 WORK3(I)=WORK7(I)                                               QAP3 247
001532  000          DO 616 I=1,20                                                   QAP3 248
001533  000      616 WORK6(I)=CMCST(I)                                               QAP3 249
001534  000    C                                                                     QAP3 250
001535  000    C  *** BACKTRACK TO THE NEXT LEVEL UP, THE TREE                       QAP3 251
001536  000    C   A) FREE THE FACILITY AT THAT LEVEL                               QAP3 252
001537  000      620 DO 640 I=1,NFCIL                                                QAP3 253
001538  000          IF(LEVEL(I).EQ.LEVY)GO TO 650                                   QAP3 254
001539  000      640 CONTINUE                                                        QAP3 255
001540  000    C                                                                     QAP3 256
001541  000      650 FCFLG(I)=.FALSE.                                                QAP3 257
001542  000          DO 670 J=1,NLOCT                                                QAP3 258
001543  000          IF(J.EQ.LOCAT(I))GO TO 680                                      QAP3 259
001544  000      670 CONTINUE                                                        QAP3 260
001545  000      680 FLOGT(J)=.FALSE.                                                QAP3 261
001546  000          IWD=-PTCST(LEVY)                                                QAP3 262
001547  000          IUBCS=IUBCS+IWD                                                 QAP3 263
001548  000          BANED(I,J)=.TRUE.                                               QAP3 264
001549  000    C                                                                     QAP3 265
001550  000    C                                                                     QAP3 266
001551  000    C   B) UPDATE THE FLAGS OF THE TWO ORDERED ARRAYS                     QAP3 267
001552  000          CALL UPDATE(I,J,1)                                              QAP3 268
001553  000    C                                                                     QAP3 269
001554  000    C   C) CALCULATE A LB ON ALL SOLUTIONS EXCLUDING THE LAST            QAP3 270
001555  000          CALL LWBND(LEVY,ITER,IWD,2)                                     QAP3 271
001556  000    C                                                                     QAP3 272
001557  000    C   D) CHECK WHETHER IT PAYS TO RESUME THE SEARCH ALONG THIS BRANCH   QAP3 273
001558  000          IF(LBCST.LT.ITOT )GO TO 730                                     QAP3 274
001559  000          KPATH=KPATH+1                                                   QAP3 275
001560  000          KEEP=I                                                          QAP3 276
001561  000          GO TO 740                                                       QAP3 277
001562  000    C                                                                     QAP3 278
001563  000    C   RETRIEVE THE INDEX OF THE FACILITY AT THAT LEVEL                 QAP3 279
001564  000      730 KEEP=FIXED(LEVY)                                                QAP3 280
```

```
001565  000          ITER=ITER+1                                                    QAP3 281
001566  000          GO TO 470                                                      QAP3 282
001567  000    C                                                                    QAP3 283
001568  000      740 DO 760 J=1,NLOCT                                               QAP3 284
001569  000      760 RANED(KEEP,J)=.FALSE.                                          QAP3 285
001570  000          FCFLG(KEEP)=.FALSE.                                            QAP3 286
001571  000          LEVY=LEVY-1                                                    QAP3 287
001572  000          ITER=ITER+1                                                    QAP3 288
001573  000          IF(ITER.LT.KOMIT)GO TO 764
001574  000          KOMIT=KOMIT+(100*NFCIL)
001575  000          KEEP=0
001576  000          DO 762 I=1,NFCIL
001577  000      762 KEEP=KEEP+IPASS(I)
001578  000          WRITE(6,52)ITER
001579  000          WRITE(6,28)(IPASS(I),I=1,NFCIL)
001580  000          WRITE(6,29)KEEP
001581  000          CALL TIMOUT
001582  000      764 IF(LEVY.NE.0.AND.ITER.LT.ITERM)GO TO 620                       QAP3 289
001583  000          IF(LEVY.E-.0.AND.ITER.LT.ITERM)GO TO 775                       QAP3 290
001584  000          WRITE(6,6)                                                     QAP3 291
001585  000          WRITE(6,39)LBTC,ITOT                                           QAP3 292
001586  000          WRITE(6,7)                                                     QAP3 293
001587  000          WRITE(6,52)ITER
001588  000          CALL TIMOUT
001589  000          DO 765 I=1,2                                                   QAP3 294
001590  000      765 WRITE(6,41)ITER                                                QAP3 295
001591  000          GO TO 810                                                      QAP3 296
001592  000    C                                                                    QAP3 297
001593  000    C   SINCE NO BETTER SOLUTION WAS FOUND , FLIP THE BOARDS , RESTORE THE  QAP3 298
001594  000    C   ARRAYS AND RETURN TO 620                                         QAP3 299
001595  000      775 IF(KSNSW(08).EQ.1)WRITE(6,36)LBTC,ITOT                         QAP3 300
001596  000          IF(KSNSW(08).EQ.1)WRITE(6,37)DIF                               QAP3 301
001597  000          IF(KTERM.EQ.100)GO TO 800
001598  000          CALL TIMOUT
001599  000          LBTC=ITOT                                                      QAP3 302
001600  000          IF(MORE.EQ.0)GO TO 800
001601  000          KELBC=LBTC                                                     QAP3 303
001602  000          IC1=KETOT                                                      QAP3 304
001603  000          IUBCS=KETOT                                                    QAP3 305
001604  000          ITOT=KETOT                                                     QAP3 306
001605  000           DIF=ITOT-LBTC                                                 QAP3 307
001606  000          IF(DIF.LT.ISTOP)GO TO 778                                      QAP3 308
001607  000           DIF=FRAC*DIF                                                  QAP3 309
001608  000          GO TO 780
001609  000    778   IF(KSNSW(08).EQ.1)WRITE(6,37)DIF                              QAP3 310
001610  000          IF(KTERM.EQ.1)GO TO 800
001611  000          KTERM=1
001612  000          ITOT=KETOT
001613  000          GO TO 782
001614  000    C                                                                    QAP3 312
001615  000      780 KETOT=ITOT                                                     QAP3 313
001616  000          ITOT=LBTC+DIF                                                  QAP3 314
001617  000      782 DO 785 I=1,1300                                                QAP3 315
001618  000          WORK8(I)=.FALSE.                                               QAP3 316
001619  000          IF(WORK2(I))WORK8(I)=.TRUE.                                    QAP3 317
001620  000      785 CONTINUE                                                       QAP3 318
001621  000          DO 790 I=1,120                                                 QAP3 319
```

```
001624      000      790 WORK7(I)=WORK3(I)                                                    QAP3 320
001625      000          DO 795 I=1,20                                                        QAP3 321
001626      000      795 CMCST(I)=WORK6(I)                                                    QAP3 322
001627      000          LEVY=NFCIL                                                           QAP3 323
001628      000      C                                                                        QAP3 324
001629      000          IF(KSNSW(08).EQ.1)WRITE(6,38)LOTC,ITOT                               QAP3 325
001630      000          GO TO 620                                                            QAP3 326
001631      000      C                                                                        QAP3 327
001632      000      800 WRITE(6,16)                                                          QAP3 328
001633      000      810 KEEP=0                                                               QAP3 329
001634      000          FTHRIO=100.*FLOAT(KPATH)/FLOAT(KLBND)                                QAP3 330
001635      000          CALL TIMOUT
001636      000          DO 820 I=1,NFCIL                                                     QAP3 331
001637      000          LIXED(I)=I                                                           QAP3 332
001638      000      820 KEEP=KEEP+IPASS(I)                                                   QAP3 333
001639      000          WRITE(6,23)                                                          QAP3 334
001640      000          WRITE(6,24)NFCIL                                                     QAP3 335
001641      000          WRITE(6,26)NLOCT                                                     QAP3 336
001642      000          WRITE(6,07)                                                          QAP3 337
001643      000          WRITE(6,47)NFIXD                                                     QAP3 338
001644      000          WRITE(6,48)                                                          QAP3 339
001645      000          DO 825 I=1,NFIXD                                                     QAP3 340
001646      000      825 WRITE(6,49)I,FIX(I),WHERE(I)                                         QAP3 341
001647      000          WRITE(6,08)                                                          QAP3 342
001648      000          DO 830 I=1,NFCIL                                                     QAP3 343
001649      000      830 WRITE(6,1)(F(I,J),J=1,NFCIL)                                         QAP3 344
001650      000          WRITE(6,05)                                                          QAP3 345
001651      000          DO 840 I=1,NLOCT                                                     QAP3 346
001652      000      840 WRITE(6,1)(D(I,J),J=1,NLOCT)                                         QAP3 347
001653      000          WRITE(6,27)ITER                                                      QAP3 348
001654      000          WRITE(6,28)(IPASS(I),I=1,NFCIL)                                      QAP3 349
001655      000          WRITE(6,29)KEEP                                                      QAP3 350
001656      000          WRITE(6,31)                                                          QAP3 351
001657      000          WRITE(6,32)(LIXED(I),I=1,NFCIL)                                      QAP3 352
001658      000          DO 850 I=1,IEVAL                                                     QAP3 353
001659      000      850 WRITE(6,33)I,ICOST(I),FOUND(I),(LOCT(I,L),L=1,NFCIL)                 QAP3 354
001660      000          WRITE(6,34)TIME                                                      QAP3 355
001661      000      C                                                                        QAP3 356
001662      000          WRITE(6,07)                                                          QAP3 357
001663      000          WRITE(6,42)                                                          QAP3 358
001664      000          WRITE(6,43)KLBND                                                     QAP3 359
001665      000          WRITE(6,44)KUPDT                                                     QAP3 360
001666      000          WRITE(6,07)                                                          QAP3 361
001667      000          WRITE(6,46)FTHRIO                                                    QAP3 362
001668      000      C                                                                        QAP3 363
001669      000      C                                                                        QAP3 364
001670      000      C   CHECK WHETHER THERE ARE ANY MORE PROBLEMS TO SOLVE                   QAP3 365
001671      000      900 IF(MORE)1000,1000,10                                                 QAP3 366
001672      000      C                                                                        QAP3 367
001673      000     1000 STOP                                                                 QAP3 368
001674      000      C                                                                        QAP3 369
001675      000          1 FORMAT(10X,40I3)                                                   QAP3 370
001676      000          2 FORMAT(10X,9('*** QAP 33 **'))                                     QAP3 371
001677      000          3 FORMAT(20X,20I3)                                                   QAP3 372
001678      000          4 FORMAT(1H1,'   DATA FOR PROBLEM',I5/                               QAP3 373
001679      000           *       10X,'      NUMBER OF FACILITIES   =',I3/                   QAP3 374
001680      000           *       10X,'      NUMBER OF LOCATIONS    =',I3//)                 QAP3 375
```

```
001679  000       5 FORMAT(10X,'    DIST...                                                    QAP3 376
001680  000       6 FORMAT(1H1)                                                                QAP3 377
001681  000       7 FORMAT(/)                                                                  QAP3 378
001682  000       8 FORMAT(/10X,'INTERACTIONS MATRIX'/)                                        QAP3 379
001683  000       9 FORMAT(20X,'AT ITERATION ****  LEVEL',I3,'  FACILITY',I3,'  IS  QAP3 380
                   *LOCATED',I3,'    WITH A ... IN COST ...',I5)                              QAP3 381
001685  000      11 FORMAT(1H0,5X,'RESULTS OF THE POST RECENT FEASIBLE SOLUTION , FOUNQAP3 382
                   *D AT ITERATION',I6/)                                                      QAP3 383
001687  000      12 FORMAT(20X,'LEVEL     FACILITY   LOCATION   PREST    CMCST'/)          QAP3 384
001688  000      14 FORMAT(22X,I2,10X,I2,9X,I2,4X,I5)                                        QAP3 385
001689  000      16 FORMAT(10X,'END OF CALCOMP PLOT.')                                       QAP3 386
001690  000      17 FORMAT(/10X,'STARTING ... ON THE TOTAL COST=',I5/)                       QAP3 387
001691  000      21 FORMAT(10X,'  ARRAY OF ... DISTANCES FOLLOWS'/)                          QAP3 388
001692  000      22 FORMAT(10X,'  ARRAY OF ... INTERACTIONS FOLLOWS')                        QAP3 389
001693  000      23 FORMAT(1H1,'  SUMMARY OF ... ITERATIONS WITH PROBLEM'/)                  QAP3 390
001694  000      24 FORMAT(10X,'NUMBER OF ... =',I5)                                         QAP3 391
001695  000      26 FORMAT(10X,'NUMBER OF ... =',I5)                                         QAP3 392
001696  000      27 FORMAT(/10X,'NUMBER OF ... =',I8)                                        QAP3 393
001697  000      28 FORMAT(/10X,'NUMBER OF ... AT VARIOUS LEVELS',8I4)                       QAP3 394
001698  000      29 FORMAT(/10X,'TOTAL ... =',I7)                                            QAP3 395
001699  000      31 FORMAT(/50X,'LOCATIONS OF ... FACILITIES')                               QAP3 396
001700  000      32 FORMAT(/10X,'NUM ,  OF ... AT',9X,2...                                   QAP3 397
001701  000      33 FORMAT(10X,I3,I7,19...                                                   QAP3 398
001702  000      34 FORMAT(//10X,'TOTAL ... =',F10.2,'  ...')                               QAP3 399
001703  000   C                                                                              QAP3 400
001704  000      36 FORMAT(/10X,'AS A RESULT ... NEWLY FOUND ... SOLUTION , WE HAVE ,  QAP3 401
                   *LATEST LO=',I5,'  ...=',I5,'  LATEST ...',I5)                           QAP3 402
001706  000      37 FORMAT(/10X,'HOWEVER , IT ... NOT SEEM TO BE WORTH IT AS DIF=',  QAP3 403
                   *1PE20.8)                                                                QAP3 404
001708  000      38 FORMAT(/15X,'THE CURRENT ... AND UPPER ... ARE ',2I7)                    QAP3 405
001709  000      39 FORMAT(/10X,'SINCE NO ... SOLUTION WAS FOUND BETWEEN',I5,' + ',QAP3 406
                   *I5,'  I WILL RESTART ... THE LAST COMPLETE SOLUTION')                   QAP3 407
001711  000      41 FORMAT(02X,3(' FORCED ... **** '),I5,' AFTER ',I8,' ITERQAP3 408
                   *ATIONS')                                                                QAP3 409
001713  000   C                                                                              QAP3 410
001714  000      42 FORMAT(10X,'NUMBER OF ... SUBROUTINE ...'/)                              QAP3 411
001715  000      43 FORMAT(10X,'SUBROUTINE ... WAS CALLED ... TIMES')                        QAP3 412
001716  000      44 FORMAT(10X,'SUBROUTINE ... WAS CALLED ... TIMES')                        QAP3 413
001717  000      46 FORMAT(10X,'FATHOMING ...',F8.0,'  PER CENT')                            QAP3 414
001718  000   C                                                                              QAP3 415
001719  000      47 FORMAT(/10X,'NUMBER OF ... ELEMENTS=',I5,'  ARRANGED AS FOLLOWS'QAP3 416
001720  000         */)                                                                     QAP3 417
001721  000      48 FORMAT(10X,'NUM',5X,'...',5X,'AT LOCATION'/)                             QAP3 418
001722  000      49 FORMAT(10X,I2,9X,I2,10...)                                               QAP3 419
001723  000   C                                                                              QAP3 420
001724  000      51 FORMAT(10X,'STARTING ... ',I8/                                           QAP3 421
001725  000         *       10X,'STARTING ... ',I6/)                                         QAP3 422
001726  000      52 FORMAT(/10X,'NO. OF ... SO FAR=',I5)                                     QAP3 423
001727  000      98 FORMAT(02X,4('  **** ERROR ****'),10X,' ...,GT.1, IT EQUALS',  QAP3 424
001728  000         *I5)                                                                     QAP3 425
001729  000      99 FORMAT(2X,5('   ***NOTICE ***'),'     IUBOS=',I5,'   ITOT=',I5)        QAP3 426
001730  000   C                                                                              QAP3 427
001731  000         END                                                                      QAP3 428
001732  000   @ELT,SI TUMER-S*SINAN-QAP5,UPDATE,,,21?330053012
001733  000         SUBROUTINE UPDATE(I,...)                                                QAP3 505
001734  000         COMMON IC1,IC2,IC3,...,N2,M2,LOCST,ITOT                                 QAP3 506
001735  000         COMMON MSTER(04000),IW...(2000),KSNSW(60)                               QAP3 507
```

```
001736  000        COMMON KLOND,KUPDT                                          QAP3 508
001737  000        INTEGER DISTI(400),DISTJ(400),INTRI(400),INTRJ(400)         QAP3 509
001738  000        LOGICAL FCFLG(20),FLOGT(20),FLGDS(400),FLGIN(400)           QAP3 510
001739  000  C                                                                 QAP3 511
001740  000        EQUIVALENCE( MSTER(0801),DISTI(1))                          QAP3 512
001741  000        EQUIVALENCE( MSTER(1201),DISTJ(1))                          QAP3 513
001742  000        EQUIVALENCE( MSTER(1601),INTRI(1))                          QAP3 514
001743  000        EQUIVALENCE( MSTER(2001),INTRJ(1))                          QAP3 515
001744  000  C                                                                 QAP3 516
001745  000        EQUIVALENCE( MSTER(2521),FCFLG(1))                          QAP3 517
001746  000        EQUIVALENCE( MSTER(2541),FLOGT(1))                          QAP3 518
001747  000        EQUIVALENCE( MSTER(2601),FLGDS(1))                          QAP3 519
001748  000        EQUIVALENCE( MSTER(3001),FLGIN(1))                          QAP3 520
001749  000  C                                                                 QAP3 521
001750  000  C                                                                 QAP3 522
001751  000  C   UPDATE THE NUMBER OF CALLS COUNTER                            QAP3 523
001752  000        KUPDT=KUPDT+1                                               QAP3 524
001753  000  C                                                                 QAP3 525
001754  000  C                                                                 QAP3 526
001755  000  C **NOTATIONS                                                     QAP3 527
001756  000  C   I    FACILITY                                                  QAP3 528
001757  000  C   J    LOCATION                                                  QAP3 529
001758  000  C   KEY= 1 ...DISENGAGEMENT                                       QAP3 530
001759  000  C   KEY= 2 ...ENGAGEMENT                                          QAP3 531
001760  000  C   UNFLAG/FLAG THE ORDERED DISTANCES AND INTERACTIONS ACCORDING TO LASTQAP3 532
001761  000  C   DISENGAGRMENT/ASSIGNMENT                                      QAP3 533
001762  000  C   A) LOCATION J HAS BEEN VACATED/ASSIGNED                       QAP3 534
001763  000        DO 280 IK=1,N2                                              QAP3 535
001764  000        II=DISTI(IK)                                                QAP3 536
001765  000        JJ=DISTJ(IK)                                                QAP3 537
001766  000        GO TO(210,240),KEY                                          QAP3 538
001767  000    210 IF(J.EQ.II.AND..NOT.FLOGT(JJ))GO TO 220                     QAP3 539
001768  000        IF(J.EQ.JJ.AND..NOT.FLOGT(II))GO TO 220                     QAP3 540
001769  000        GO TO 280                                                   QAP3 541
001770  000    220 FLGDS(IK)=.FALSE.                                           QAP3 542
001771  000        GO TO 280                                                   QAP3 543
001772  000    240 IF(J.EQ.II.OR.J.EQ.JJ)GO TO 260                             QAP3 544
001773  000        GO TO 280                                                   QAP3 545
001774  000    260 FLGDS(IK)=.TRUE.                                            QAP3 546
001775  000    280 CONTINUE                                                    QAP3 547
001776  000  C                                                                 QAP3 548
001777  000  C   B) FACILITY I HAS BEEN DISENTANGLED/LOCATED                   QAP3 549
001778  000        JK=J                                                        QAP3 550
001779  000        DO 380 J=1,M2                                               QAP3 551
001780  000        II=INTRI(J)                                                 QAP3 552
001781  000        JJ=INTRJ(J)                                                 QAP3 553
001782  000        GO TO(310,340),KEY                                          QAP3 554
001783  000    310 IF(I.EQ.II.AND..NOT.FCFLG(JJ))GO TO 320                     QAP3 555
001784  000        IF(I.EQ.JJ.AND..NOT.FCFLG(II))GO TO 320                     QAP3 556
001785  000        GO TO 380                                                   QAP3 557
001786  000    320 FLGIN(J)=.FALSE.                                            QAP3 558
001787  000        GO TO 380                                                   QAP3 559
001788  000    340 IF(I.EQ.II. OR.I.EQ.JJ)GO TO 360                            QAP3 560
001789  000        GO TO 380                                                   QAP3 561
001790  000    360 FLGIN(J)=.TRUE.                                             QAP3 562
001791  000    380 CONTINUE                                                    QAP3 563
001792  000        J=JK                                                        QAP3 564
```

```
001795  000   C                                                                      QAP3 565
001794  000         RETURN                                                           QAP3 566
001795  000         END                                                              QAP3 567
001796  000   @ELT,SI TUMER-S*SINAN-QAP5,LWBND,,,212650053012
001797  000   C   A SUBROUTINE TO CALCULATE THE LOWER BOUND AT ANY NODE              QAP3 568
001798  000         SUBROUTINE LWBND(LEVY,ITER,IWD,MOVE)                             QAP3 569
001799  000         COMMON IC1,IC2,IC3,NFCIL,NLOCT,N2,M2,LRCST,ITOT                  QAP3 570
001800  000         COMMON MSTER(04000),IWORK(2000),KSNSW(60)                        QAP3 571
001801  000         COMMON KLBND,KUPDT                                               QAP3 572
001802  000         INTEGER F(20,20),D(20,20),LOCAT(20),LEVEL(20)                    QAP3 573
001803  000         INTEGER WORK1(20,20),WORK2(20),WORK3(20),INDEX(20),JNDEX(20)     QAP3 574
001804  000         INTEGER DIST1(400),DIST2(400),ITRY(400),IDUM(400)                QAP3 575
001805  000         LOGICAL FCFLG(20),FLAG1(20),BADBD(20,20)                         QAP3 576
001806  000         LOGICAL FLGD5(400),FLGD6(400)                                    QAP3 577
001807  000   C                                                                      QAP3 578
001808  000         EQUIVALENCE( MSTER(0001),F(1,1))                                 QAP3 579
001809  000         EQUIVALENCE( MSTER(0401),D(1,1))                                 QAP3 580
001810  000         EQUIVALENCE( MSTER(0801),LOCAT(1))                               QAP3 581
001811  000         EQUIVALENCE( MSTER(0821),LEVEL(1))                               QAP3 582
001812  000         EQUIVALENCE( MSTER(1501),DIST1(1))                               QAP3 583
001813  000         EQUIVALENCE( MSTER(1901),DIST2(1))                               QAP3 584
001814  000   C                                                                      QAP3 585
001815  000         EQUIVALENCE( MSTER(2301),FLAG1(1))                               QAP3 586
001816  000         EQUIVALENCE( MSTER(2321),BADBD(1,1))                             QAP3 587
001817  000         EQUIVALENCE( MSTER(2721),ITRY(1))                                QAP3 588
001818  000         EQUIVALENCE( MSTER(3121),FLGD5(1))                               QAP3 589
001819  000   C                                                                      QAP3 590
001820  000         EQUIVALENCE( MSTER(3521),FLGD6(1))                               QAP3 591
001821  000         EQUIVALENCE( MSTER(3921),IDUM(1))                                QAP3 592
001822  000         EQUIVALENCE( MSTER(4321),WORK1(1,1))                             QAP3 593
001823  000   C                                                                      QAP3 594
001824  000         EQUIVALENCE( IWORK(1),MSTER(0001))                               QAP3 595
001825  000   C                                                                      QAP3 596
001826  000   C   UPDATE THE NUMBER OF CALLS COUNTER                                QAP3 597
001827  000         KLBND=KLBND+1                                                    QAP3 598
001828  000   C                                                                      QAP3 599
001829  000         IF(KSNSW(21).EQ.1.AND.MOVE.EQ.0)WRITE(3,10)                      QAP3 600
001830  000         IF(KSNSW(21).EQ.1.AND.MOVE.EQ.0)WRITE(3,15)(I,I=1,LEVY)          QAP3 601
001831  000         IF(KSNSW(21).EQ.1.AND.MOVE.EQ.0)WRITE(3,21)(LOCAT(I),I=1,LEVY    QAP3 602
001832  000   C                                                                      QAP3 603
001833  000         IC1=IC1+IWD                                                      QAP3 604
001834  000   C                                                                      QAP3 605
001835  000         IF(KSNSW(22).EQ.1)WRITE(6,22)IC1                                 QAP3 606
001836  000   C                                                                      QAP3 607
001837  000   C   CALCULATE THE IC2 MATRIX                                          QAP3 608
001838  000     260 IROW=0                                                           QAP3 609
001839  000         IC2=0                                                            QAP3 610
001840  000         IF(MOVE.NE.0)GO TO 270                                           QAP3 611
001841  000         IROW=NFCIL                                                       QAP3 612
001842  000         GO TO 580                                                        QAP3 613
001843  000     270 DO 400 I=1,NFCIL                                                 QAP3 614
001844  000         IF(FCFLG(I))GO TO 400                                            QAP3 615
001845  000         IROW=IROW+1                                                      QAP3 616
001846  000         WORK2(IROW)=32767                                                QAP3 617
001847  000         ICOL=0                                                           QAP3 618
001848  000         DO 350 J=1,NLOCT                                                 QAP3 619
001849  000         IF(FLOGT(J))GO TO 350                                            QAP3 620
```

```
001850  000          ICOL=ICOL+1                                                QAP3 621
001851  000          IF(.NOT.BANED(I,J))GO TO 400                               QAP3 622
001852  000          KEEP=32767                                                 QAP3 623
001853  000          GO TO 340                                                  QAP3 624
001854  000      280 KEEP=0                                                     QAP3 625
001855  000          DO 330 K=1,NFCIL                                           QAP3 626
001856  000          IF(.NOT.FCFLG(K))GO TO 330                                 QAP3 627
001857  000          L=LOCAT(K)                                                 QAP3 628
001858  000          ISAV=F(I,K)                                                QAP3 629
001859  000          IF(F(K,I).GT.F(I,K))ISAV=F(K,I)                            QAP3 630
001860  000          KEEP=KEEP+(ISAV*D(J,L))                                    QAP3 631
001861  000      330 CONTINUE                                                   QAP3 632
001862  000      340 WORK1(IROW,ICOL)=KEEP                                      QAP3 633
001863  000          IF(KEEP.GE.WORK2(IROW))GO TO 350                           QAP3 634
001864  000          WORK2(IROW)=KEEP                                           QAP3 635
001865  000      350 CONTINUE                                                   QAP3 636
001866  000          IC2=IC2+WORK2(IROW)                                        QAP3 637
001867  000      400 CONTINUE                                                   QAP3 638
001868  000      C                                                              QAP3 639
001869  000      C   WRITE THE IC2 MATRIC IF YOU WISH                           QAP3 640
001870  000          IF(KSNSW(23).EQ.0)GO TO 460                                QAP3 641
001871  000          KEEP=0                                                     QAP3 642
001872  000          DO 430 I=1,NFCIL                                           QAP3 643
001873  000          IF(FCFLG(I))GO TO 430                                      QAP3 644
001874  000          KEEP=KEEP+1                                                QAP3 645
001875  000          INDEX(KEEP)=I                                              QAP3 646
001876  000      430 CONTINUE                                                   QAP3 647
001877  000          KEEP=0                                                     QAP3 648
001878  000          DO 450 J=1,NLOCT                                           QAP3 649
001879  000          IF(FLOGT(J))GO TO 450                                      QAP3 650
001880  000          KEEP=KEEP+1                                                QAP3 651
001881  000          JNDEX(KEEP)=J                                              QAP3 652
001882  000      450 CONTINUE                                                   QAP3 653
001883  000          WRITE(6,23)                                                QAP3 654
001884  000          WRITE(6,11)(JNDEX(J),J=1,ICOL)                             QAP3 655
001885  000          DO 460 I=1,IROW                                            QAP3 656
001886  000      460 WRITE(6,1)INDEX(I),(WORK1(I,J),J=1,ICOL),WORK2(I)          QAP3 657
001887  000          WRITE(6,7)                                                 QAP3 658
001888  000      C   CALCULATE THE MODIFIED MATRIX AND GET COLUMN MINIMA         QAP3 659
001889  000      480 DO 500 J=1,ICOL                                            QAP3 660
001890  000          WORK3(J)=32767                                             QAP3 661
001891  000          DO 490 I=1,IROW                                            QAP3 662
001892  000          WORK1(I,J)=WORK1(I,J)-WORK2(I)                             QAP3 663
001893  000          IF(WORK1(I,J).GE.WORK3(J))GO TO 490                        QAP3 664
001894  000          WORK3(J)=WORK1(I,J)                                        QAP3 665
001895  000      490 CONTINUE                                                   QAP3 666
001896  000      500 CONTINUE                                                   QAP3 667
001897  000      C                                                              QAP3 668
001898  000      C   PRINT COLUMN MINIMA                                        QAP3 669
001899  000          IF(KSNSW(23).EQ.0)GO TO 530                                QAP3 670
001900  000          WRITE(6,11)(WORK3(J),J=1,ICOL)                             QAP3 671
001901  000      C                                                              QAP3 672
001902  000      C   IN CASE ICOL.GT.IROW TAKE THE FIRST IROW COLUMNS AFTER RANKING QAP3 673
001903  000      530 IF(ICOL.EQ.IROW)GO TO 540                                  QAP3 674
001904  000          CALL ARRNG(ICOL,1,1)                                       QAP3 675
001905  000      540 DO 550 J=1,IROW                                            QAP3 676
001906  000      550 IC2=IC2+WORK3(J)                                           QAP3 677
```

```
001907      000    C                                                                  QAP3 678
001908      000    C   WRITE RHE VALUES OF IC2 IF YOU WISH                            QAP3 679
001909      000      IF(KSNSW(24).EQ.1)WRITE(6,24)IC2                                 QAP3 680
001910      000    C                                                                  QAP3 681
001911      000    C                                                                  QAP3 682
001912      000    C   CALCULATE THE VALUE OD IC3                                     QAP3 683
001913      000     580 IC3=0                                                         QAP3 684
001914      000         IF(IROW.EQ.1)GO TO 700                                        QAP3 685
001915      000         IF(KSNSW(26).EQ.1)WRITE(6,26)                                 QAP3 686
001916      000         KEEP=0                                                        QAP3 687
001917      000         K=1                                                           QAP3 688
001918      000         IROW2=IROW*(IROW-1)/2                                         QAP3 689
001919      000         DO 650 I=1,N2                                                 QAP3 690
001920      000         IF(FLGDS(I))GO TO 650                                         QAP3 691
001921      000         DO 620 J=K,M2                                                 QAP3 692
001922      000         IF(FLGIN(J))GO TO 620                                         QAP3 693
001923      000         KEEP=KEEP+1                                                   QAP3 694
001924      000         II=DISTI(I)                                                   QAP3 695
001925      000         IJ=DISTJ(I)                                                   QAP3 696
001926      000         JI=INTRI(J)                                                   QAP3 697
001927      000         JJ=INTRJ(J)                                                   QAP3 698
001928      000         KAM=D(II,IJ)*F(JI,JJ)                                         QAP3 699
001929      000         IC3=IC3+KAM                                                   QAP3 700
001930      000         IF(KSNSW(26).EQ.1)WRITE(6,13)KEEP,II,IJ,JI,JJ,D(II,IJ),F(JI,JJ)  QAP3 701
001931      000        *,KAM                                                          QAP3 702
001932      000         GO TO 630                                                     QAP3 703
001933      000     620 CONTINUE                                                      QAP3 704
001934      000     630 IF(KEEP.E-.IROW2)GO TO 710                                    QAP3 705
001935      000         K=J+1                                                         QAP3 706
001936      000     650 CONTINUE                                                      QAP3 707
001937      000         DO 660 I=1,10                                                 QAP3 708
001938      000     660 WRITE(6,98)KEEP,IROW,IROW2                                    QAP3 709
001939      000         STOP                                                          QAP3 710
001940      000    C                                                                  QAP3 711
001941      000     700 IF(KSNSW(27).EQ.1)WRITE(6,19)IC3                              QAP3 712
001942      000         IF(MOVE.NE.0)GO TO 720                                        QAP3 713
001943      000         KEEP=IC1+IC2+IC3                                              QAP3 714
001944      000         IF(KEEP.GT.LBCST)GO TO 720                                    QAP3 715
001945      000         WRITE(6,27)KEEP,LBCST                                         QAP3 716
001946      000         GO TO 750                                                     QAP3 717
001947      000     720 LBCST=IC1+IC2+IC3                                             QAP3 718
001948      000     750 IF(KSNSW(28).EQ.1)WRITE(6,29)LBCST                            QAP3 719
001949      000         IF(KSNSW(29).EQ.1)WRITE(6,17)IC1,IC2,IC3,LBCST                QAP3 720
001950      000    C                                                                  QAP3 721
001951      000    C                                                                  QAP3 722
001952      000       1 FORMAT(10X,20I5)                                             QAP3 723
001953      000       7 FORMAT(/)                                                    QAP3 724
001954      000      11 FORMAT(15X,20I5/)                                            QAP3 725
001955      000      12 FORMAT(12X,I2,6X,I2,3X,I2,3X,I2,3(7X,I4))                    QAP3 726
001956      000      14 FORMAT(90X,'VALUE OF C3 =',I7/)                              QAP3 727
001957      000      16 FORMAT(90X,'LOWER BOUND =',I7//)                             QAP3 728
001958      000      17 FORMAT(/50X,'C1=',I5,5X,'C2=',I5,5X,'C3=',I5,5X,'LBCST=',I7) QAP3 729
001959      000      18 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT THE START') QAP3 730
001960      000      19 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT ITER ',I3,' QAP3 731
001961      000        *LEVEL',I3,' ON A FORWARD MOVE')                              QAP3 732
001962      000
001963      000
```

```
001964      000
001965      000      21 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT ITER ',I3,'      QAP3 733
001966      000        *LEVEL',I3,' ON A BACKWARD MOVE')                                    QAP3 734
001967      000      22 FORMAT(90X,'VALUE IF C1 =',I5/)                                     QAP3 735
001968      000      23 FORMAT(10X,'MATRIX FOR CALCULATING C2 FOLLOWS'/)                    QAP3 736
001969      000      24 FORMAT(90X,'VALUE OF C2 =',I5/)                                     QAP3 737
001970      000      26 FORMAT(10X,'ARRAY OF ELEMENTS TO CALCULATE C3 FOLLOWS'//,           QAP3 738
001971      000        *      10X,'INDEX    FROM/TO    INTER BET.    DIST.    INTER    VAL QAP3 739
001972      000        *UE')                                                                QAP3 740
001973      000      27 FORMAT(/10X,'CALCULATED STARTING LB=',I5,'   SO WE TAKE LBCST=',    QAP3 741
001974      000        *I5)                                                                 QAP3 742
001975      000      98 FORMAT(02X,20(1H*),3(' ERROR IN LWBND $$'),'  KEEP=',I3,'  IROW=QAP3 743
001976      000        *',I3,'  IROW2=',I3)                                                 QAP3 744
001977      000         RETURN                                                            QAP3 745
001978      000         END                                                               QAP3 746
001979      000   @ELT,SI TUMER-S*SINAN-QAP5.MAX,,,213106053012
001980      000         FUNCTION MAX(K,L)                                                 QAP3 747
001981      000         MAX=K                                                             QAP3 748
001982      000         IF(L.GT.K)MAX=L                                                   QAP3 749
001983      000         RETURN                                                            QAP3 750
001984      000         END                                                               QAP3 751
001985      000   @ELT,SI TUMER-S*SINAN-QAP5,/BRN,,,,155634053112
001986      000   C    A SUBROUTINE TO ARRANGE THE ARRAY                                    QAP3 429
001987      000         SUBROUTINE ARRNG(N,KEY,KOLY)                                      QAP3 430
001988      000         COMMON IC1,IC2,IC3,NFCIL,NLOCT,M2,M2,LBCST,ITOT                   QAP3 431
001989      000         COMMON MSTER(04000),IWORK1(4000),KSNSW(60)                        QAP3 432
001990      000         COMMON KLPHD,KUPDT                                                QAP3 433
001991      000         INTEGER F(20,20),D(20,20),DISTI(400),DISTJ(400),INTRI(400)        QAP3 434
001992      000         INTEGER INTRJ(400)                                                QAP3 435
001993      000         INTEGER VISTI(400),VISTJ(400),VALUE(400),WORK3(20)                QAP3 436
001994      000         EQUIVALENCE( MSTER(0001),F(1,1))                                  QAP3 437
001995      000         EQUIVALENCE( MSTER(0401),D(1,1))                                  QAP3 438
001996      000         EQUIVALENCE( MSTER(0801),DISTI(1))                                QAP3 439
001997      000         EQUIVALENCE( MSTER(1201),DISTJ(1))                                QAP3 440
001998      000         EQUIVALENCE( MSTER(1601),INTRI(1))                                QAP3 441
001999      000         EQUIVALENCE( MSTER(2001),INTRJ(1))                                QAP3 442
002000      000   C                                                                       QAP3 443
002001      000         EQUIVALENCE(IWORK1(0001),VISTI(001))                              QAP3 444
002002      000         EQUIVALENCE(IWORK1(0401),VISTJ(001))                              QAP3 445
002003      000         EQUIVALENCE(IWORK1(1201),VALUE(001))                              QAP3 446
002004      000         EQUIVALENCE(IWORK1(1980),WORK3(001))                              QAP3 447
002005      000   C                                                                       QAP3 448
002006      000         IF(KONY.EQ.1)KEEP=N                                               QAP3 449
002007      000         IF(KONY.EQ.1)GO TO 110                                            QAP3 450
002008      000         NNN=NFCIL                                                         QAP3 451
002009      000         IF(KEY.EQ.1)NNN=NLOCT                                             QAP3 452
002010      000         KEEP=0                                                            QAP3 453
002011      000         DO 100 I=1,NNN                                                    QAP3 454
002012      000         DO 100 J=1,NNN                                                    QAP3 455
002013      000         KEEP=KEEP+1                                                       QAP3 456
002014      000         IF(KEY.EQ.1)VALUE(KEEP)=D(I,J)                                    QAP3 457
002015      000         IF(KEY.EQ.2)VALUE(KEEP)=F(I,J)                                    QAP3 458
002016      000         IF(KEY.EQ.1.AND.VALUE(KEEP).EQ.0)VALUE(KEEP)=32767                QAP3 459
002017      000         IF(KEY.EQ.1.AND.I.GT.J)VALUE(KEEP)=32767                          QAP3 460
002018      000         VISTI(KEEP)=I                                                     QAP3 461
002019      000       100 VISTJ(KEEP)=J                                                   QAP3 462
002020      000         GO TO 120                                                         QAP3 463
```

```
002021      000      110 DO 115 I=1,N                                                    QAP3 464
002022      000      115 VALUE(I)=WORK3(I)                                               QAP3 465
002023      000      120 N1=N-1                                                          QAP3 466
002024      000          DO 300 I=1,N1                                                   QAP3 467
002025      000          J1=I+1                                                          QAP3 468
002026      000          DO 200 J=J1,N                                                   QAP3 469
002027      000          GO TO(130,150),KEY                                              QAP3 470
002028      000      130 IF(VALUE(I).LE.VALUE(J))GO TO 200                               QAP3 471
002029      000          GO TO 160                                                       QAP3 472
002030      000      150 IF(VALUE(I).GE.VALUE(J))GO TO 200                               QAP3 473
002031      000      160 KEEP=VALUE(I)                                                   QAP3 474
002032      000          VALUE(I)=VALUE(J)                                               QAP3 475
002033      000          VALUE(J)=KEEP                                                   QAP3 476
002034      000          IF(KONY.EQ.1)GO TO 200                                          QAP3 477
002035      000      C                                                                   QAP3 478
002036      000          KEEP=VISTI(I)                                                   QAP3 479
002037      000          VISTI(I)=VISTI(J)                                               QAP3 480
002038      000          VISTI(J)=KEEP                                                   QAP3 481
002039      000      C                                                                   QAP3 482
002040      000          KEEP=VISTJ(I)                                                   QAP3 483
002041      000          VISTJ(I)=VISTJ(J)                                               QAP3 484
002042      000          VISTJ(J)=KEEP                                                   QAP3 485
002043      000      200 CONTINUE                                                        QAP3 486
002044      000      300 CONTINUE                                                        QAP3 487
002045      000      C                                                                   QAP3 488
002046      000          GO TO (380,310),KONY                                            QAP3 489
002047      000
002048      000
002049      000
002050      000          GO TO(320,340),KEY                                              QAP3 491
002051      000      320 DISTI(I)=VISTI(I)                                               QAP3 492
002052      000          DISTJ(I)=VISTJ(I)                                               QAP3 493
002053      000          GO TO 350                                                       QAP3 494
002054      000      340 INTRI(I)=VISTI(I)                                               QAP3 495
002055      000          INTRJ(I)=VISTJ(I)                                               QAP3 496
002056      000      350 CONTINUE                                                        QAP3 497
002057      000          GO TO 500                                                       QAP3 498
002058      000      C                                                                   QAP3 499
002059      000      380 DO 390 I=1,N                                                    QAP3 500
002060      000      390 WORK3(I)=VALUE(I)                                               QAP3 501
002061      000      C                                                                   QAP3 502
002062      000      500 RETURN                                                          QAP3 503
002063      000          END                                                             QAP3 504
002064      000      QELT,SI TUMER-S*SINAN-QAP5.TIMOUT,,,163213053112
002065      000          SUBROUTINE TIMOUT
002066      000          COMMON IC1,IC2,IC3,NFCTL,NLOCT,N2,M2,LPCST,ITOT
002067      000          COMMON MSTER(04000),IWORK(2000),PSNSW(60)
002068      000          COMMON KLBND,KUPDT
002069      000          K=IITIME(I,J)
002070      000          T=I/5000.
002071      000          WRITE(6,2000)T
002072      000       2000 FORMAT(' TIME',F10.3,' SEC')
002073      000          RETURN
002074      000          END
002075      000      QELT,SI TUMER-S*SINAN-QAP5.DTAIN,,,163704053112
002076      000      C    A SUBROUTINE TO READ DATA FOR THE QUADRATIC ASSIGNMENT PROBLEM  QAP3 752
002077      000      C    THIS VERSION IS INTENDED FOR THE NUGENT ET AL DATA              QAP3 753
```

```
002078  000        SUBROUTINE DTAIN(JOB,IFRAC,NFIXD,ISTOP,MORE,ITERM)                    QAP3 754
002079  000        COMMON IC1,IC2,IC3,NFCIL,NLOCT,N2,M2,LBCST,ITOT                       QAP3 755
002080  000        COMMON MSTER(04000),IWORK(2000),KSNSW(60)                             QAP3 756
002081  000        COMMON KLRND,KUPDT                                                    QAP3 757
002082  000        INTEGER   F(20,20),D(20,20),FIX(20),WHERE(20)                         QAP3 758
002083  000         INTEGER WORK1(20,20)
002084  000        EQUIVALENCE( MSTER(0001),F(1,1))                                      QAP3 760
002085  000        EQUIVALENCE( MSTER(0401),D(1,1))                                      QAP3 761
002086  000        EQUIVALENCE( MSTER(3201),WHERE(1))                                    QAP3 762
002087  000        EQUIVALENCE( MSTER(3241),  FIX(1))                                    QAP3 763
002088  000        EQUIVALENCE( IWORK(0001),WORK1(1,1))                                  QAP3 764
002089  000   C                                                                          QAP3 765
002090  000   C   READ THE NUMBER OF FACILITIES , THE NUMBER OF LOCATIONS , THE FRACTI QAP3 766
002091  000   C   OF THE INTERVAL (ITOT-LBTC) , THE NUMBER OF FIXED FACILITIES , AND   QAP3 767
002092  000   C   THE CONTINUATION PARAMETER                                           QAP3 768
002093  000        READ(5,1)JOB,NFCIL,NLOCT,IFRAC,NFIXD,MORE,ISTOP,ITERM,LBCST,ITOT     QAP3 769
002094  000   C                                                                          QAP3 770
002095  000        IF(ITOT.EQ.0)ITOT=3276700                                             QAP3 771
002096  000        DO 10 I=1,20                                                          QAP3 772
002097  000        DO 10 J=1,20                                                          QAP3 773
002098  000        F(I,J)=0                                                              QAP3 774
002099  000     10 D(I,J)=0                                                              QAP3 775
002100  000   C                                                                          QAP3 776
002101  000        DO 20 I=1,NFCIL                                                       QAP3 777
002102  000     20 READ(5,2)(WORK1(I,J),J=1,NFCIL)                                       QAP3 778
002103  000   C                                                                          QAP3 779
002104  000        DO 30 I=1,NFCIL                                                       QAP3 780
002105  000        DO 25 J=1,NFCIL                                                       QAP3 781
002106  000        IF(I.EQ.J)GO TO 25                                                    QAP3 782
002107  000        IF(I.GT.J)F(I,J)=WORK1(I,J)                                           QAP3 783
002108  000        IF(I.LT.J)D(I,J)=WORK1(I,J)                                           QAP3 784
002109  000     25 CONTINUE                                                              QAP3 785
002110  000     30 CONTINUE                                                              QAP3 786
002111  000   C                                                                          QAP3 787
002112  000        DO 40 I=1,NFCIL                                                       QAP3 788
002113  000        DO 40 J=1,NFCIL                                                       QAP3 789
002114  000        F(I,J)=F(J,I)                                                         QAP3 790
002115  000        D(J,I)=D(I,J)                                                         QAP3 791
002116  000     40 CONTINUE                                                              QAP3 792
002117  000   C                                                                          QAP3 793
002118  000        DO 50 I=1,NFCIL                                                       QAP3 794
002119  000        DO 45 J=1,NFCIL                                                       QAP3 795
002120  000        IF(I.GT.J)F(I,J)=0                                                    QAP3 796
002121  000     45 CONTINUE                                                              QAP3 797
002122  000     50 CONTINUE                                                              QAP3 798
002123  000   C                                                                          QAP3 799
002124  000        KSNSW(1)=1                                                            QAP3 800
002125  000        KSNSW(4)=1                                                            QAP3 801
002126  000        KSNSW(6)=1                                                            QAP3 802
002127  000        KSNSW(07)=1                                                           QAP3 803
002128  000        KSNSW(08)=1                                                           QAP3 804
002129  000   C   READ THE INDICES OF THE FACILITIES FIXED APRIORI                      QAP3 805
002130  000        DO 70 I=1,NFIXD                                                       QAP3 806
002131  000     70 READ(5,1)  FIX(I),WHERE(I)                                           QAP3 807
002132  000   C                                                                          QAP3 808
002133  000        NFIXD=1                                                               QAP3 809
002134  000        RETURN                                                                QAP3 810
```

```
      C                                                                      QAP3 811
      C                                                                      QAP3 812
      C                                                                      QAP3 813
          1 FORMAT(20X,12I5)                                                 QAP3 814
          2 FORMAT(10X,35I2)                                                 QAP3 815
            END                                                              QAP3 816
      @ELT,SI TUMER-S*SINAN-QAP4,M                                 7060512
            COMMON IC1,IC2,IC3,N                                             QAP4   1
            COMMON MSTER(04000),I                                            QAP4   2
            COMMON KLOND,KUPDT                                               QAP4   3
            INTEGER F(20,20),W(20,20),DIST(400),DIST1(400),INTRI(400)        QAP4   4
            INTEGER INTRJ(400),                                              QAP4   5
            INTEGER FIXED(20),                                               QAP4   6
            INTEGER IPASS(20),                                               QAP4   7
            INTEGER  FIX(20),                                                QAP4   8
            LOGICAL FCFLG(20),                                               QAP4   9
      C                                                                      QAP4  10
            LOGICAL WORK2(1300),                                             QAP4  11
            INTEGER WORK3(0120),                                             QAP4  12
      C                                                                      QAP4  13
      C                                                                      QAP4  14
            EQUIVALENCE(                                                     QAP4  15
            EQUIVALENCE(                                                     QAP4  16
            EQUIVALENCE(                                                     QAP4  17
            EQUIVALENCE(                                                     QAP4  18
            EQUIVALENCE(                                                     QAP4  19
            EQUIVALENCE(                                                     QAP4  20
            EQUIVALENCE(                                                     QAP4  21
            EQUIVALENCE(                                                     QAP4  22
            EQUIVALENCE(                                                     QAP4  23
            EQUIVALENCE(                                                     QAP4  24
            EQUIVALENCE(                                                     QAP4  25
            EQUIVALENCE(                                                     QAP4  26
            EQUIVALENCE(                                                     QAP4  27
            EQUIVALENCE(                                                     QAP4  28
            EQUIVALENCE(                                                     QAP4  29
            EQUIVALENCE(                                                     QAP4  30
            EQUIVALENCE(                                                     QAP4  31
            EQUIVALENCE(                                                     QAP4  32
            EQUIVALENCE(                                                     QAP4  33
      C                                                                      QAP4  34
            EQUIVALENCE( IWORK(0001),                                        QAP4  35
      C                                                                      QAP4  36
            EQUIVALENCE( MSTER(2001),                                        QAP4  37
            EQUIVALENCE( MSTER(2021),                                        QAP4  38
      C                                                                      QAP4  39
      C   READ THE KEYS SETTINGS FOR THE MAIN PROGRAM                        QAP4  40
            READ(5,3)(KSNSW(I),I=01,20)                                      QAP4  41
      C   READ THE KEY SETTINGS FOR SUBROUTINE LWBND                        QAP4  42
            READ(5,3)(KSNSW(I),I=21,30)                                      QAP4  43
      C                                                                      QAP4  44
      C   READ THE PROBLEM DATA                                             QAP4  45
          10 CALL DTAIN(JOB,IFRAC,NFCIL,ISTOP,MORE,ITERM)                    QAP4  46
      C                                                                      QAP4  47
      C   PRINT THE INITIAL DATA IF YOU WISH                                QAP4  48
            IF(KSNSW(01).EQ.0)GO TO 110                                      QAP4  49
            WRITE(6,4)JOB,NFCIL,NLOCT                                        QAP4  50
```

```
002192  000          DO 70 I=1,2                                          QAP4  51
002193  000       70 WRITE(6,2)                                           QAP4  52
002194  000          WRITE(6,7)                                           QAP4  53
002195  000          WRITE(6,5)                                           QAP4  54
002196  000          DO 80 I=1,NLOCT                                      QAP4  55
002197  000       80 WRITE(6,01)(D(I,J),J=1,NLOCT)                        QAP4  56
002198  000          WRITE(6,8)                                           QAP4  57
002199  000          DO 90 I=1,NFCIL                                      QAP4  58
002200  000       90 WRITE(6,01)(F(I,J),J=1,NFCIL)                        QAP4  59
002201  000          WRITE(6,7)                                           QAP4  60
002202  000          WRITE(6,51)LBCST,ITOT                                QAP4  61
002203  000    C   INITIALIZE THE COST                                    QAP4  62
002204  000      110 IC1=0                                                QAP4  63
002205  000          N2=NLOCT*NLOCT                                       QAP4  64
002206  000          M2=NFCIL*NFCIL                                       QAP4  65
002207  000    C   UNFLAG ALL THE LOCATIONS AND THE FACILITIES            QAP4  66
002208  000          DO 130 I=1,NLOCT                                     QAP4  67
002209  000      130 FLOGT(I)=.FALSE.                                     QAP4  68
002210  000          DO 140 J=1,NFCIL                                     QAP4  69
002211  000      140 FCFLG(J)=.FALSE.                                     QAP4  70
002212  000    C                                                          QAP4  71
002213  000          DO 150 I=1,NFCIL                                     QAP4  72
002214  000          IPASS(I)=0                                           QAP4  73
002215  000          DO 150 J=1,NLOCT                                     QAP4  74
002216  000      150 BANED(I,J)=.FALSE.                                   QAP4  75
002217  000    C   RANK THE DISTANCES IN AN ASCENDING ORDER              QAP4  76
002218  000          CALL ARRNG(N2,1,2)                                   QAP4  77
002219  000    C   PRINT THR MATRIX OF ARRANGED DISTANCES                 QAP4  78
002220  000          IF(KSNSW(02).EQ.0)GO TO 200                          QAP4  79
002221  000          WRITE(6,21)                                          QAP4  80
002222  000          WRITE(6,01)(DISTI(I),I=1,N2)                         QAP4  81
002223  000          WRITE(6,07)                                          QAP4  82
002224  000          WRITE(6,01)(DISTJ(I),I=1,N2)                         QAP4  83
002225  000    C                                                          QAP4  84
002226  000    C   FLAG ALL THE DISTANCES                                 QAP4  85
002227  000      200 DO 210 I=1,N2                                        QAP4  86
002228  000      210 FLGDS(I)=.FALSE.                                     QAP4  87
002229  000    C                                                          QAP4  88
002230  000    C   RANK THE INTERACTIONS IN A DESCENDING ORDER           QAP4  89
002231  000          CALL ARRNG(M2,2,2)                                   QAP4  90
002232  000    C   PRINT THE INTERACTIONS IF YOU WISH                     QAP4  91
002233  000          IF(KSNSW(03).EQ.0)GO TO 310                          QAP4  92
002234  000          WRITE(6,22)                                          QAP4  93
002235  000          WRITE(6,01)(INTRI(I),I=1,M2)                         QAP4  94
002236  000          WRITE(6,07)                                          QAP4  95
002237  000          WRITE(6,01)(INTRJ(I),I=1,M2)                         QAP4  96
002238  000    C                                                          QAP4  97
002239  000    C   FLAGG ALL THE INTERACTIONS                            QAP4  98
002240  000      310 DO 320 I=1,M2                                        QAP4  99
002241  000      320 FLGIN(I)=.FALSE.                                     QAP4 100
002242  000    C   INITIALIZE WORK7                                       QAP4 101
002243  000          DO 325 I=1,120                                       QAP4 102
002244  000      325 WORK7(I)=0                                           QAP4 103
002245  000    C   INITIALIZE WORK8                                       QAP4 104
002246  000          DO 330 I=1,1300                                      QAP4 105
002247  000      330 WORK8(I)=.FALSE.                                     QAP4 106
002248  000          DO 335 I=1,20                                        QAP4 107
```

```
002249  000      355 CMCST(I)=0                                                GAP4 108
002250  000    C                                                               GAP4 109
002251  000    C                                                               GAP4 110
002252  000          TIME=0                                                    GAP4 111
002253  000          LBTC=LBCST                                                GAP4 112
002254  000          IEVAL=0                                                   GAP4 113
002255  000          IUBCS=0                                                   GAP4 114
002256  000          KOMPL=0                                                   GAP4 115
002257  000          FRAC=IFRAC                                                GAP4 116
002258  000          FRAC=0.01*FRAC                                            GAP4 117
002259  000    C                                                               GAP4 118
002260  000    C   INITIALIZE COUNTERS                                         GAP4 119
002261  000          KLBND=0                                                   GAP4 120
002262  000          KUPDT=0                                                   GAP4 121
002263  000          KPATH=0                                                   GAP4 122
002264  000          KTERM=0
002265  000    C                                                               GAP4 123
002266  000    C                                                               GAP4 124
002267  000    C                                                               GAP4 125
002268  000    C   A) MAKE THE FIXED ASSIGNMENTS FIRST                         GAP4 126
002269  000          ITER=1                                                    GAP4 127
002270  000          NFXID=NFIXD                                               GAP4 128
002271  000          LEVY=1                                                    GAP4 129
002272  000          PICST(1)=0                                                GAP4 130
002273  000          IF(NFIXD.EQ.1)GO TO 425                                   GAP4 131
002274  000          DO 420 I=2,NFIXD                                          GAP4 132
002275  000          L2=  FIX(I)                                               GAP4 133
002276  000           J=WHERE(I)                                               GAP4 134
002277  000          LEVEL(L2)=999                                             GAP4 135
002278  000          LOCAT(L2)=J                                               GAP4 136
002279  000          FCFLG(L2)=.TRUE.                                          GAP4 137
002280  000          FLOGT( J)=.TRUE.                                          GAP4 138
002281  000          DO 400 JJ=1,NLOCT                                         GAP4 139
002282  000          IF(JJ.EQ.J)GO TO 400                                      GAP4 140
002283  000          BANED(L2,JJ)=.TRUE.                                       GAP4 141
002284  000      400 CONTINUE                                                  GAP4 142
002285  000          DO 405 JJ=1,NFCIL                                         GAP4 143
002286  000          IF(JJ.EQ.L2)GO TO 405                                     GAP4 144
002287  000          BANED(JJ,J)=.TRUE.                                        GAP4 145
002288  000      405 CONTINUE                                                  GAP4 146
002289  000          CALL UPDATE(L2,J,2)                                       GAP4 147
002290  000          IF(KSNSW(05).EQ.1)WRITE(6,9)ITER,LEVY,L2,J                GAP4 148
002291  000          DO 410 JET=2,NFIXD                                        GAP4 149
002292  000          IF(JET.EQ.I)GO TO 410                                     GAP4 150
002293  000          JJ=WHERE(JET)                                             GAP4 151
002294  000          LL=FIX(JET)                                               GAP4 152
002295  000          ISAV=F(L2,LL)                                             GAP4 153
002296  000          IF(F(LL,L2).GT.ISAV)ISAV=F(LL,L2)                         GAP4 154
002297  000          IC1=IC1+(ISAV*D(J,JJ))                                    GAP4 155
002298  000      410 CONTINUE                                                  GAP4 156
002299  000      420 CONTINUE                                                  GAP4 157
002300  000          IC1=0.5*IC1                                               GAP4 158
002301  000          IUBCS=IUBCS+IC1                                           GAP4 159
002302  000    C                                                               GAP4 160
002303  000    C   CALCULATE A STARTING LOWER BOUND                           GAP4 161
002304  000          CALL LWBND(0,0,0,1)                                       GAP4 162
002305  000          GO TO 426                                                 GAP4 163
```

```
002306  000  C                                                                      GAP4 164
002307  000    425 CALL LWBND(0,0,0,0)                                              GAP4 165
002308  000  C                                                                      GAP4 166
002309  000    426 IF(LBTC.LT.LBCST)LBTC=LBCST                                      GAP4 167
002310  000        IF(KSNSW(04).EQ.1)WRITE(6,17)LBTC                                GAP4 168
002311  000  C                                                                      GAP4 169
002312  000  C   B)PROCEED TO OBTAIN A FIRST FEASIBLE SOLUTION ALONG THE RIGHTMOST  GAP4 170
002313  000  C   BRANCH OR THE TREE                                                 GAP4 171
002314  000        L2=FIX(1)                                                        GAP4 172
002315  000        J=WHERE(1)                                                       GAP4 173
002316  000        LEVEL(L2)=1                                                      GAP4 174
002317  000        FIXED(1)=L2                                                      GAP4 175
002318  000        LOCAT(L2)=J                                                      GAP4 176
002319  000        LIXED(1)=J                                                       GAP4 177
002320  000        JET=0                                                            GAP4 178
002321  000        IF(NFIXD.EQ.1)GO TO 429                                          GAP4 179
002322  000        DO 427 I=2,NFIXD                                                 GAP4 180
002323  000        JJ=WHERE(I)                                                      GAP4 181
002324  000        LL=  FIX(I)                                                      GAP4 182
002325  000        ISAV=F(L2,LL)                                                    GAP4 183
002326  000        IF(ISAV.LT.F(LL,L2))ISAV=F(LL,L2)                                GAP4 184
002327  000        JET=JET+(ISAV*D(J,JJ))                                           GAP4 185
002328  000    427 CONTINUE                                                         GAP4 186
002329  000    429 PTCST(1)=JET                                                     GAP4 187
002330  000        CMCST(1)=JET+IC1                                                 GAP4 188
002331  000        IUBCS=IUBCS+JET                                                  GAP4 189
002332  000        FCLTY(1)=L2                                                      GAP4 190
002333  000        FCFLG(L2)=.TRUE.                                                 GAP4 191
002334  000        FLOGT(J)=.TRUE.                                                  GAP4 192
002335  000        L1=J                                                             GAP4 193
002336  000        CALL UPDATE(L2,L1,2)                                             GAP4 194
002337  000  C   B) CHOOSE THE ELEMENT WITH MAXIMUM INTERACTIONS WITH THE ONE ASSIGNEGAP4 195
002338  000  C   AT THE PRECEEDING LEVEL                                           GAP4 196
002339  000    450 LEVY=LEVY+1                                                      GAP4 197
002340  000        IPASS(LEVY)=IPASS(LEVY)+1                                        GAP4 198
002341  000        ITER=ITER+1                                                      GAP4 199
002342  000        JET=FCLTY(LEVY-1)                                                GAP4 200
002343  000        ISAV=-32767                                                      GAP4 201
002344  000        DO 460 I=1,NFCIL                                                 GAP4 202
002345  000        IF(FCFLG(I))GO TO 460                                            GAP4 203
002346  000        IF(F(I,JET).LE.ISAV.AND.F(JET,I).LE.ISAV)GO TO 460               GAP4 204
002347  000        ISAV=F(I,JET)                                                    GAP4 205
002348  000        IF(F(JET,I).GT.F(I,JET))ISAV=F(JET,I)                            GAP4 206
002349  000        KEEP=I                                                           GAP4 207
002350  000    460 CONTINUE                                                         GAP4 208
002351  000    470 FCFLG(KEEP)=.TRUE.                                               GAP4 209
002352  000        LEVEL(KEEP)=LEVY                                                 GAP4 210
002353  000  C   C) ASSIGN THE ELEMENT TO A LOCATION SO THAT THE COST OF SUCH AN ASSIGAP4 211
002354  000  C     IS MINIMAL                                                       GAP4 212
002355  000    510 IWD=32767                                                        GAP4 213
002356  000        DO 530 J=1,NLOCT                                                 GAP4 214
002357  000        IF(BANED(KEEP,J))GO TO 530                                       GAP4 215
002358  000        IF(FLOGT(J))GO TO 530                                            GAP4 216
002359  000        KAM=0                                                            GAP4 217
002360  000        DO 520 I=1,NFCIL                                                 GAP4 218
002361  000        IF(I.EQ.KEEP)GO TO 520                                           GAP4 219
002362  000        IF(.NOT.FCFLG(I))GO TO 520                                       GAP4 220
```

```
002365      000          L=LOCAT(I)                                                    QAP4 221
002364      000          ISAVV=F(KEEP,I)                                               QAP4 222
002365      000          IF(F(I,KEEP).GT.F(KEEP,I))ISAVV=F(I,KEEP)                     QAP4 223
002366      000          KAM=KAM*(ISAVV*D(J,L))                                        QAP4 224
002367      000      520 CONTINUE                                                      QAP4 225
002368      000          IF(KAM.GT.IWD)GO TO 530                                       QAP4 226
002369      000          IWD=KAM                                                       QAP4 227
002370      000          LOCAT(KEEP)=J                                                 QAP4 228
002371      000      530 CONTINUE                                                      QAP4 229
002372      000          IF(IWD.EQ.32767)GO TO 740                                     QAP4 230
002373      000          IF(KSNSW(05).EQ.1)WRITE(6,9)ITER,LEVY,KEEP,LOCAT(KEEP),IWD    QAP4 231
002374      000          J=LOCAT(KEEP)                                                 QAP4 232
002375      000          FCLTY(LEVY)=KEEP                                              QAP4 233
002376      000          FLOGT(J)=.TRUE.                                               QAP4 234
002377      000          IUBCS=IUBCS+IWD                                               QAP4 235
002378      000          FIXED(LEVY)=KEEP                                              QAP4 236
002379      000          PTCST(LEVY)=IWD                                               QAP4 237
002380      000          CALL UPDATE(KEEP,J,2)                                         QAP4 238
002381      000          IF(KSNSW(06).EQ.0)GO TO 560                                   QAP4 239
002382      000          CMCST(LEVY)=IUBCS                                             QAP4 240
002383      000          LIXED(LEVY)=J                                                 QAP4 241
002384      000      C                                                                 QAP4 242
002385      000      C   CHECK WHETHER WE HAVE REACHED A TERMINAL NODE OR NOT         QAP4 243
002386      000      580 LEVX=LEVY+NFIXD-1                                             QAP4 244
002387      000          IF( LEVX.LT.NFCIL.AND.IWD.NE.0)GO TO 630                      QAP4 245
002388      000          IF( LEVX.EQ.NFCIL)GO TO 590                                   QAP4 246
002389      000      C   CHECK WHETHER IT PAYS TO CONTINUE THE SEARCH ALONG THIS BRANCH QAP4 247
002390      000          CALL LWBND(LEVY,ITER,IWD,1)                                   QAP4 248
002391      000          IF(LBCST.LT.ITOT)GO TO 585                                    QAP4 249
002392      000          KPATH=KPATH+1                                                 QAP4 250
002393      000          GO TO 620                                                     QAP4 251
002394      000      585 LEVY=LEVY+1                                                   QAP4 252
002395      000          IPASS(LEVY)=IPASS(LEVY)+1                                     QAP4 253
002396      000          GO TO 730                                                     QAP4 254
002397      000      C   E) NOW WE HAVE A COMPLETE SOLUTION , PRINT IT IF YOU WISH     QAP4 255
002398      000      590 KOMPL=1                                                       QAP4 256
002399      000          IC1=IUBCS                                                     QAP4 257
002400      000          IF(IUBCS.GE.ITOT)GO TO 630                                    QAP4 258
002401      000          ITOT=IUBCS                                                    QAP4 259
002402      000          IEVAL=IEVAL+1                                                 QAP4 260
002403      000          ICOST(IEVAL)=IUBCS                                            QAP4 261
002404      000          FOUND(IEVAL)=ITER                                             QAP4 262
002405      000          DO 595 I=1,NFCIL                                              QAP4 263
002406      000          ILOCT(IEVAL,I)=LOCAT(I)                                       QAP4 264
002407      000      595 CONTINUE                                                      QAP4 265
002408      000      600 IF(KSNSW(06).EQ.0)GO TO 610                                   QAP4 266
002409      000          WRITE(6,11)ITER                                               QAP4 267
002410      000          WRITE(6,12)                                                   QAP4 268
002411      000          DO 605 I=1,LEVY                                               QAP4 269
002412      000      605 WRITE(6,14)I,FIXED(I),LIXED(I),PTCST(I),CMCST(I)              QAP4 270
002413      000          CALL TIMOUT
002414      000      C                                                                 QAP4 271
002415      000      C                                                                 QAP4 272
002416      000      C   REDUCE THE CURRENT LOWER BOUND                               QAP4 273
002417      000      610 DIF=ITOT-LBTC                                                 QAP4 274
002418      000          KETOT=ITOT                                                    QAP4 275
002419      000          KELBC=LBTC                                                    QAP4 276
```

```
002420      000          IF(DIF.LE.ISTOP)KTER=100
002421      000          IF(DIF.LE.ISTOP)GO TO 611                                  QAP4 277
002422      000          DIF=(FRAC*DIF)                                             QAP4 278
002423      000          ITOT=LBTC+DIF                                              QAP4 279
002424      000        611 IF(KSNSW(08).EQ.1)WRITE(6,350)KCLRC,ITOT,KETOT
002425      000    C                                                                QAP4 281
002426      000    C     F)STORE THE CONFIGURATION OF THE CURRENT NODE SO THAT WE CAN RESTORFQAP4 282
002427      000    C       IT IF WE FAIL TO OBTAIN A BETTER SOLUTION UNDER THE CURRENT CEILINQAP4 283
002428      000          DO 612 I=1,1300                                            QAP4 284
002429      000          WORK2(I)=.FALSE.                                           QAP4 285
002430      000          IF(WORK8(I))WORK2(I)=.TRUE.                                QAP4 286
002431      000        612 CONTINUE                                                 QAP4 287
002432      000          DO 615 I=1,120                                             QAP4 288
002433      000        615 WORK3(I)=WORK7(I)                                        QAP4 289
002434      000          DO 616 I=1,20                                              QAP4 290
002435      000        616 WORK6(I)=CMCST(I)                                        QAP4 291
002436      000    C                                                                QAP4 292
002437      000    C   *** BACKTRACK TO THE NEXT LEVEL UP, THE TREE                 QAP4 293
002438      000    C     A) FREE THE FACILITY AT THAT LEVEL                         QAP4 294
002439      000        620 DO 640 I=1,NFCIL                                         QAP4 295
002440      000          IF(LEVEL(I).EQ.LEVY)GO TO 650                              QAP4 296
002441      000        640 CONTINUE                                                 QAP4 297
002442      000    C                                                                QAP4 298
002443      000        650 FCFLG(I)=.FALSE.                                         QAP4 299
002444      000          DO 670 J=1,NLOCT                                           QAP4 300
002445      000          IF(J.EQ.LOCAT(I))GO TO 660                                 QAP4 301
002446      000        670 CONTINUE                                                 QAP4 302
002447      000        660 FLOGT(J)=.FALSE.                                         QAP4 303
002448      000          IWD=-PTCST(LEVY)                                           QAP4 304
002449      000          IUBCS=IUBCS+IWD                                            QAP4 305
002450      000          BANED(I,J)=.TRUE.                                          QAP4 306
002451      000    C                                                                QAP4 307
002452      000    C                                                                QAP4 308
002453      000    C     B) UPDATE THE FLAGS OF THE TWO ORDERED ARRAYS              QAP4 309
002454      000          CALL UPDATE(I,J,1)                                         QAP4 310
002455      000    C                                                                QAP4 311
002456      000    C     C) CALCULATE A LB ON ALL SOLUTIONS EXCLUDING THE LAST      QAP4 312
002457      000          CALL LWBND(LEVY,ITER,IWD,0)                                QAP4 313
002458      000    C                                                                QAP4 314
002459      000    C     D) CHECK WHETHER IT PAYS TO RESUME THE SEARCH ALONG THIS BRANCQAP4 315
002460      000          IF(LBCST.LT.ITOT )GO TO 750                                QAP4 316
002461      000          KPATH=KPATH+1                                              QAP4 317
002462      000          KEEP=I                                                     QAP4 318
002463      000          GO TO 740                                                  QAP4 319
002464      000    C                                                                QAP4 320
002465      000    C     RETRIEVE THE INDEX OF THE FACILITY AT THAT LEVEL           QAP4 321
002466      000        750 KEEP=FIXED(LEVY)                                         QAP4 322
002467      000          ITER=ITER+1                                                QAP4 323
002468      000          GO TO 470                                                  QAP4 324
002469      000    C                                                                QAP4 325
002470      000        740 DO 760 J=1,NLOCT                                         QAP4 326
002471      000        760 BANED(KEEP,J)=.FALSE.                                    QAP4 327
002472      000          FCFLG(KEEP)=.FALSE.                                        QAP4 328
002473      000          LEVY=LEVY-1                                                QAP4 329
002474      000          ITER=ITER+1                                                QAP4 330
002475      000          IF(LEVY.NE.0.AND.ITER.LT.ITERM)GO TO 620                   QAP4 331
002476      000          IF(LEVY.EQ.0.AND.ITER.LT.ITERM)GO TO 775                   QAP4 332
```

```
002477      000          WRITE(6,6)                                                    QAP4 333
002478      000          WRITE(6,39)LBTC,ITOT                                          QAP4 334
002479      000          WRITE(6,7)                                                    QAP4 335
002480      000          DO 765 I=1,2                                                  QAP4 336
002481      000        765 WRITE(6,41)ITER                                             QAP4 337
002482      000          GO TO 810                                                     QAP4 338
002483      000    C                                                                   QAP4 339
002484      000    C   SINCE NO BETTER SOLUTION WAS FOUND , FLIP THE BOUNDS , RESTORE THE  QAP4 340
002485      000    C   ARRAYS AND RETURN TO 620                                        QAP4 341
002486      000        775 IF(KSNSW(08).EQ.1)WRITE(6,39)LBTC,ITOT                      QAP4 342
002487      000            IF(KSNSW(08).EQ.1)WRITE(6,42)ITER                           QAP4 343
002488      000            CALL TIMOUT
002489      000            IF(KTERM.EQ.100)GO TO 780
002490      000            LBTC=ITOT                                                   QAP4 344
002491      000            KELBC=LBTC                                                  QAP4 345
002492      000            IC1=KETOT                                                   QAP4 346
002493      000            IUBCS=KETOT                                                 QAP4 347
002494      000            ITOT=KETOT                                                  QAP4 348
002495      000             DIF=ITOT-LBTC                                              QAP4 349
002496      000            IF(DIF.LT.ISTOP)GO TO 778                                   QAP4 350
002497      000             DIF=FRAC*DIF                                               QAP4 351
002498      000            GO TO 780
002499      000        778   IF(KSNSW(08).EQ.1)WRITE(6,37)DIF                          QAP4 352
002500      000            IF(KTERM.EQ.1)GO TO 780
002501      000            KTERM=1
002502      000            ITOT=KETOT
002503      000            GO TO 782
002504      000    C                                                                   QAP4 354
002505      000        780 KETOT=ITOT                                                  QAP4 355
002506      000            ITOT=LBTC+DIF                                               QAP4 356
002507      000        782 DO 785 I=1,1300                                             QAP4 357
002508      000            WORK8(I)=.FALSE.                                            QAP4 358
002509      000            IF(WORK2(I))WORK8(I)=.TRUE.                                 QAP4 359
002510      000        785 CONTINUE                                                    QAP4 360
002511      000            DO 790 I=1,120                                              QAP4 361
002512      000        790 WORK7(I)=WORK3(I)                                           QAP4 362
002513      000            DO 795 I=1,20                                               QAP4 363
002514      000        795 CMCST(I)=WORK6(I)                                           QAP4 364
002515      000            LEVY=NFCIL-NFIXD+1                                          QAP4 365
002516      000    C                                                                   QAP4 366
002517      000            IF(KSNSW(08).EQ.1)WRITE(6,38)LBTC,ITOT                      QAP4 367
002518      000            GO TO 620                                                   QAP4 368
002519      000    C                                                                   QAP4 369
002520      000        800 WRITE(6,16)                                                 QAP4 370
002521      000        810 KEEP=0                                                      QAP4 371
002522      000            FTHRIO=100.*FLOAT(KFATH)/FLOAT(KLBND)                       QAP4 372
002523      000            DO 820 I=1,NFCIL                                            QAP4 373
002524      000            LIXED(I)=I                                                  QAP4 374
002525      000        820 KEEP=KEEP+IPASS(I)                                          QAP4 375
002526      000            WRITE(6,23)                                                 QAP4 376
002527      000            WRITE(6,24)NFCIL                                            QAP4 377
002528      000            WRITE(6,26)NLOCT                                            QAP4 378
002529      000            WRITE(6,07)                                                 QAP4 379
002530      000            WRITE(6,47)NFIXD                                            QAP4 380
002531      000            WRITE(6,48)                                                 QAP4 381
002532      000            DO 825 I=1,NFIXD                                            QAP4 382
002533      000        825 WRITE(6,49)I,FIX(I),WHERE(I)                                QAP4 383
```

```
002534      000          WRITE(6,08)                                                    QAP4 384
002535      000          DO 830 I=1,NFCIL                                               QAP4 385
002536      000      830 WRITE(6,1)(F(I,J),J=1,NFCIL)                                   QAP4 386
002537      000          WRITE(6,05)                                                    QAP4 387
002538      000          DO 840 I=1,NLOCT                                               QAP4 388
002539      000      840 WRITE(6,1)(D(I,J),J=1,NLOCT)                                   QAP4 389
002540      000          WRITE(6,27)ITCR                                                QAP4 390
002541      000          WRITE(6,28)(IPASS(I),I=1,NFCTL)                                QAP4 391
002542      000          WRITE(6,29)KEEP                                                QAP4 392
002543      000          WRITE(6,31)                                                    QAP4 393
002544      000          WRITE(6,32)(LIXED(I),I=1,NFCTL)                                QAP4 394
002545      000          DO 850 I=1,IEVAL                                               QAP4 395
002546      000      850 WRITE(6,33)I,ICOST(I),FOUND(I),(TLOCT(I,L),L=1,NFCIL)          QAP4 396
002547      000          WRITE(6,34)TIME                                                QAP4 397
002548      000      C                                                                  QAP4 398
002549      000          WRITE(6,07)                                                    QAP4 399
002550      000          WRITE(6,42)                                                    QAP4 400
002551      000          WRITE(6,43)KLBND                                               QAP4 401
002552      000          WRITE(6,44)KUPDT                                               QAP4 402
002553      000          WRITE(6,07)                                                    QAP4 403
002554      000          WRITE(6,46)FTHRIO                                              QAP4 404
002555      000      C                                                                  QAP4 405
002556      000      C                                                                  QAP4 406
002557      000      C   CHECK WHETHER THERE ARE ANY MORE PROBLEMS TO SOLVE             QAP4 407
002558      000          IF(MORE)1000,1000,10                                           QAP4 408
002559      000      C                                                                  QAP4 409
002560      000       1000 STOP                                                         QAP4 410
002561      000      C                                                                  QAP4 411
002562      000          1 FORMAT(10X,40I3)                                             QAP4 412
002563      000          2 FORMAT(10X,9('*** QAP 4 ***'))                               QAP4 413
002564      000          3 FORMAT(20X,20I3)                                             QAP4 414
002565      000          4 FORMAT(1H1,'   DATA FOR PROBLEM',I5/                         QAP4 415
002566      000           *       10X,'      NUMBER OF FACILITIES   =',I3/              QAP4 416
002567      000           *       10X,'      NUMBER OF LOCATIONS    =',I3//)            QAP4 417
002568      000          5 FORMAT(10X,'    DISTANCE MATRIX'/)                           QAP4 418
002569      000          6 FORMAT(1H1)                                                  QAP4 419
002570      000          7 FORMAT(/)                                                    QAP4 420
002571      000          8 FORMAT(/10X,'INTERACTION MATRIX'/)                           QAP4 421
002572      000          9 FORMAT(20X,'AT ITERATION ',I3,'  LEVEL',I3,'  FACILITY',I3,'  IS  QAP4 422
002573      000           *LOCATED',I3,'    WITH AN INCREASE IN COST EQUALS',I5)       QAP4 423
002574      000         11 FORMAT(1H0,5X,'RESULTS OF THE MOST RECENT FEASIBLE SOLUTION , FOUNQAP4 424
002575      000           *D AT ITERATION',I6/)                                        QAP4 425
002576      000         12 FORMAT(20X,'LEVEL     FACILITY   LOCATION   FTCST    CMCST'/)  QAP4 426
002577      000         14 FORMAT(22X,I2,10X,I2,9X,I3,9X,I4,4X,I6)                     QAP4 427
002578      000         16 FORMAT(10X,'END OF CALCULATIONS')                           QAP4 428
002579      000         17 FORMAT(/10X,'STARTING LOWER BOUND ON THE TOTAL COST=',I5/)  QAP4 429
002580      000         21 FORMAT(10X,'  ARRAY OF ASSIGNED DISTANCES FOLLOWS'/)        QAP4 430
002581      000         22 FORMAT(10X,'  ARRAY OF ASSIGNED INTERACTIONS FOLLOWS')      QAP4 431
002582      000         23 FORMAT(1H1,'  SUMMARY OF COMPUTATIONAL EXPERIENCE WITH PROBLEM'/) QAP4 432
002583      000         24 FORMAT(10X,'NUMBER OF FACILITIES =',I5)                     QAP4 433
002584      000         26 FORMAT(10X,'NUMBER OF LOCATIONS  =',I5)                     QAP4 434
002585      000         27 FORMAT(/10X,'NUMBER OF ITERATIONS =',I8)                    QAP4 435
002586      000         28 FORMAT(/10X,'NUMBER OF NODES AT VARIOUS LEVELS',20I4)       QAP4 436
002587      000         29 FORMAT(/10X,'TOTAL NUMBER OF NODES =',I8)                   QAP4 437
002588      000         31 FORMAT(/50X,'LOCATIONS OF THE FACILITIES')                  QAP4 438
002589      000         32 FORMAT( 10X,'NUM ,  COST',' FOUND AT',2X,20I3/)             QAP4 439
002590      000         33 FORMAT(10X,I3,I7,I9,3X,2013)                                QAP4 440
```

```
002591  000      34 FORMAT(//10X,'TOTAL CPU TIME =',F10.2,'  SECONDS')                    QAP4 441
002592  000  C                                                                           QAP4 442
002593  000      36 FORMAT(/10X,'AS A RESULT OF THIS NEWLY FOUND SOLUTION , WE HAVE ',   QAP4 443
002594  000        *'LATEST LB=',I5,'   CURRENT UB=',I5,'  LATEST BEST',I5)              QAP4 444
002595  000      37 FORMAT(/10X,'HOWEVER , IT DOES NOT SEEM WORTHWHILE TO SAVE IT AS DIF=',QAP4 445
002596  000        *1PE20.8)                                                           QAP4 446
002597  000      38 FORMAT(/15X,'THE CURRENT LOWER AND UPPER INDICES ARE ',2I7)         QAP4 447
002598  000      39 FORMAT(/10X,'SINCE NO BETTER SOLUTION WAS FOUND BETWEEN',I5,' + ',QAP4 448
002599  000        *I5,'  I WILL RESTART FROM THE LAST CONTINUATION')                   QAP4 449
002600  000      41 FORMAT(02X,3(' FOLLOWING INFORMATION *** '),10X,'  AFTER ',I8,' ITERQAP4 450
002601  000        *ATIONS')                                                           QAP4 451
002602  000  C                                                                           QAP4 452
002603  000      42 FORMAT(10X,'NUMBER OF CALLS OF SUBROUTINE ',I7)                     QAP4 453
002604  000      43 FORMAT(10X,'SUBROUTINE ',I2,' WAS CALLED ',I5,'  TIMES')            QAP4 454
002605  000      44 FORMAT(10X,'SUBROUTINE ',I2,' WAS CALLED ',I5,'  TIMES')            QAP4 455
002606  000      46 FORMAT(10X,'FATHOMING ',A6,'=',F5.1,'  PER CENT')                   QAP4 456
002607  000  C                                                                           QAP4 457
002608  000      47 FORMAT(/10X,'NUMBER OF FIXED ELEMENTS=',I5,'  ARRANGED AS FOLLOWS'QAP4 458
002609  000        */)                                                                 QAP4 459
002610  000      48 FORMAT(10X,'NUM',5X,'ELEMENT',9X,'AT LOCATION'/)                    QAP4 460
002611  000      49 FORMAT(10X,I2,9X,I2)                                                QAP4 461
002612  000  C                                                                           QAP4 462
002613  000      51 FORMAT(10X,'STARTING LOWER BOUND=',I8/                              QAP4 463
002614  000        *      10X,'STARTING UPPER BOUND=',I8/)                              QAP4 464
002615  000      52 FORMAT(/10X,'NO. OF ITERATIONS TO FARE=',I5)                        QAP4 465
002616  000  C                                                                           QAP4 466
002617  000      99 FORMAT(2X,5('   ***ERROR***'),'     IOUT=',I5,'   ITOT=',I5)       QAP4 467
002618  000  C                                                                           QAP4 468
002619  000         END                                                                 QAP4 469
002620  000  @ELT,SI TUNER-S*SINAN-QAP4.FOR,,,
```

```
002646  000            KEEP=KEEP+1                                                    QAP4 497
002649  000            IF(KEY.EQ.1)VALUE(KEEP)=DIST(I,J)                              QAP4 498
002650  000            IF(KEY.EQ.2)VALUE(KEEP)=DIST(I,J)                              QAP4 499
002651  000            IF(KEY.EQ.1.AND.VALUE(KEEP).EQ.0)VALUE(KEEP)=32767             QAP4 500
002652  000            IF(KEY.EQ.1.AND.I.GT.J)VALUE(KEEP)=32767                       QAP4 501
002653  000            VISTI(KEEP)=I                                                  QAP4 502
002654  000          100 VISTJ(KEEP)=J                                                QAP4 503
002655  000            GO TO 120                                                      QAP4 504
002656  000          110 DO 115 I=1,N                                                 QAP4 505
002657  000          115 VALUE(I)=WORK3(I)                                            QAP4 506
002658  000          120 N1=N-1                                                       QAP4 507
002659  000            DO 300 I=1,N1                                                  QAP4 508
002660  000            J1=I+1                                                         QAP4 509
002661  000            DO 200 J=J1,N                                                  QAP4 510
002662  000            GO TO(130,150),KEY                                             QAP4 511
002663  000          130 IF(VALUE(I).LE.VALUE(J))GO TO 200                            QAP4 512
002664  000            GO TO 160                                                      QAP4 513
002665  000          150 IF(VALUE(I).GE.VALUE(J))GO TO 200                            QAP4 514
002666  000          160 KEEP=VALUE(I)                                                QAP4 515
002667  000            VALUE(I)=VALUE(J)                                              QAP4 516
002668  000            VALUE(J)=KEEP                                                  QAP4 517
002669  000            IF(KONY.EQ.1)GO TO 200                                         QAP4 518
002670  000          C                                                                QAP4 519
002671  000            KEEP=VISTI(I)                                                  QAP4 520
002672  000            VISTI(I)=VISTI(J)                                              QAP4 521
002673  000            VISTI(J)=KEEP                                                  QAP4 522
002674  000          C                                                                QAP4 523
002675  000            KEEP=VISTJ(I)                                                  QAP4 524
002676  000            VISTJ(I)=VISTJ(J)                                              QAP4 525
002677  000            VISTJ(J)=KEEP                                                  QAP4 526
002678  000          200 CONTINUE                                                     QAP4 527
002679  000          300 CONTINUE                                                     QAP4 528
002680  000          C                                                                QAP4 529
002681  000            GO TO (380,310),KONY                                           QAP4 530
002682  000          310 DO 350 I=1,N                                                 QAP4 531
002683  000            GO TO(320,340),KEY                                             QAP4 532
002684  000          320 DISTI(I)=VISTI(I)                                            QAP4 533
002685  000            DISTJ(I)=VISTJ(I)                                              QAP4 534
002686  000            GO TO 350                                                      QAP4 535
002687  000          340 INTRI(I)=VISTI(I)                                            QAP4 536
002688  000            INTRJ(I)=VISTJ(I)                                              QAP4 537
002689  000          350 CONTINUE                                                     QAP4 538
002690  000            GO TO 500                                                      QAP4 539
002691  000          C                                                                QAP4 540
002692  000          380 DO 390 I=1,N                                                 QAP4 541
002693  000          390 WORK3(I)=VALUE(I)                                            QAP4 542
002694  000          C                                                                QAP4 543
002695  000          500 RETURN                                                       QAP4 544
002696  000            END                                                            QAP4 545
002697  000          @ELT,SI TUMER-S*SINAN-QAP4.UPDATE,,,234317053012
002698  000            SUBROUTINE UPDATE(I,J,KEY)                                     QAP4 546
002699  000            COMMON IC1,IC2,IC3,NFCIL,NLOCT,N2,M2,LOCST,ITOT                QAP4 547
002700  000            COMMON MSTER(04000),IWORK(2000),KSNSW(60)                      QAP4 548
002701  000            COMMON KLAND,KUPDT                                             QAP4 549
002702  000            INTEGER DISTI(400),DISTJ(400),INTRI(400),INTRJ(400)            QAP4 550
002703  000            LOGICAL FCFLG(20),FLOGT(20),FLGDS(400),FLGIN(400)              QAP4 551
002704  000          C                                                                QAP4 552
```

```
002705      000          EQUIVALENCE( MSTER(      ),DISTI(1))                                  QAP4 553
002706      000          EQUIVALENCE( MSTER(      ),DISTJ(1))                                  QAP4 554
002707      000          EQUIVALENCE( MSTER(      ),INTRI(1))                                  QAP4 555
002708      000          EQUIVALENCE( MSTER(      ),INTRJ(1))                                  QAP4 556
002709      000    C                                                                            QAP4 557
002710      000          EQUIVALENCE( MSTER(      ),FCFLG(1))                                  QAP4 558
002711      000          EQUIVALENCE( MSTER(      ),FLGT(1))                                   QAP4 559
002712      000          EQUIVALENCE( MSTER(      ),FLGDS(1))                                  QAP4 560
002713      000          EQUIVALENCE( MSTER(      ),FLGIN(1))                                  QAP4 561
002714      000    C                                                                            QAP4 562
002715      000    C   UPDATE THE COUNTER FOR THE NUMBER OF CALLS                               QAP4 563
002716      000          KUPDT=KUPDT+1                                                        QAP4 564
002717      000    C                                                                            QAP4 565
002718      000    C                                                                            QAP4 566
002719      000    C **NOTATIONS                                                              QAP4 567
002720      000    C   I    FACILITY                                                          QAP4 568
002721      000    C   J    LOCATION                                                          QAP4 569
002722      000    C   KEY= 1 ...DISENGAGEMENT                                                QAP4 570
002723      000    C   KEY= 2 ...ENGAGEMENT                                                   QAP4 571
002724      000    C   UNFLAG/FLAG THE ORDERED DISTANCES AND INTERACTIONS ACCORDING TO LAST  QAP4 572
002725      000    C   DISENGAGEMENT/ASSIGNMENT                                              QAP4 573
002726      000    C   A) LOCATION J HAS BEEN VACATED/ASSIGNED                               QAP4 574
002727      000          DO 280 IK=1,N2                                                       QAP4 575
002728      000          II=DISTI(IK)                                                         QAP4 576
002729      000          JJ=DISTJ(IK)                                                         QAP4 577
002730      000          GO TO(210,240),KEY                                                   QAP4 578
002731      000      210 IF(J.EQ.II.AND..NOT.FLGT(JJ))GO TO 220                               QAP4 579
002732      000          IF(J.EQ.JJ.AND..NOT.FLGT(II))GO TO 220                               QAP4 580
002733      000          GO TO 280                                                            QAP4 581
002734      000      220 FLGDS(IK)=.FALSE.                                                    QAP4 582
002735      000          GO TO 280                                                            QAP4 583
002736      000      240 IF(J.EQ.II.OR.J.EQ.JJ)GO TO 260                                      QAP4 584
002737      000          GO TO 280                                                            QAP4 585
002738      000      260 FLGDS(IK)=.TRUE.                                                     QAP4 586
002739      000      280 CONTINUE                                                             QAP4 587
002740      000    C                                                                            QAP4 588
002741      000    C   B) FACILITY I HAS BEEN DISENTANGLED/LOCATED                           QAP4 589
002742      000          JK=J                                                                 QAP4 590
002743      000          DO 380 J=1,M2                                                        QAP4 591
002744      000          II=INTRI(J)                                                          QAP4 592
002745      000          JJ=INTRJ(J)                                                          QAP4 593
002746      000          GO TO(310,340),KEY                                                   QAP4 594
002747      000      310 IF(I.EQ.II.AND..NOT.FCFLG(JJ))GO TO 320                              QAP4 595
002748      000          IF(I.EQ.JJ.AND..NOT.FCFLG(II))GO TO 320                              QAP4 596
002749      000          GO TO 380                                                            QAP4 597
002750      000      320 FLGIN(J)=.FALSE.                                                     QAP4 598
002751      000          GO TO 380                                                            QAP4 599
002752      000      340 IF(I.EQ.II. OR.I.EQ.JJ)GO TO 360                                     QAP4 600
002753      000          GO TO 380                                                            QAP4 601
002754      000      360 FLGIN(J)=.TRUE.                                                      QAP4 602
002755      000      380 CONTINUE                                                             QAP4 603
002756      000          J=JK                                                                 QAP4 604
002757      000    C                                                                            QAP4 605
002758      000          RETURN                                                               QAP4 606
002759      000          END                                                                  QAP4 607
002760      000    @ELT,SI TUMER-S*SINAN-QAP4.LGO,,,235321053012
002761      000    C   A SUBROUTINE TO CALCULATE THE LOWER BOUND AT ANY NODE                  QAP4 608
```

```
002762      000          SUBROUTINE LWBND(LEVY,[illegible],MOVE)                                  QAP4 609
002763      000          COMMON IC1,IC2,IC3,[illegible],N2,M2,LRCST,ITOT                          QAP4 610
002764      000          COMMON MSTER(04000),[illegible],KSNSW(60)                                QAP4 611
002765      000          COMMON KLBND,KUPDT                                                       QAP4 612
002766      000          INTEGER F(20,20),[illegible],LEVEL(20)                                   QAP4 613
002767      000          INTEGER WORK1(20,20)[illegible]RK3(20),[illegible](20),JNDEX(20)         QAP4 614
002768      000          INTEGER DIST1(400),[illegible]RI(400),IS[illegible]400)                  QAP4 615
002769      000          LOGICAL FCFLG(20),FL[illegible](20,20)                                   QAP4 616
002770      000          LOGICAL FLGDS(400),FLG[illegible]00)                                     QAP4 617
002771      000    C                                                                              QAP4 618
002772      000          EQUIVALENCE( MSTER(000[illegible]),F(1,1))                               QAP4 619
002773      000          EQUIVALENCE( MSTER([illegible]),D(1,1))                                  QAP4 620
002774      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 621
002775      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 622
002776      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 623
002777      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 624
002778      000    C                                                                              QAP4 625
002779      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 626
002780      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 627
002781      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 628
002782      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 629
002783      000    C                                                                              QAP4 630
002784      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 631
002785      000          EQUIVALENCE( MSTER([illegible]),[illegible](1))                          QAP4 632
002786      000          EQUIVALENCE( MSTER([illegible]),[illegible](1,1))                        QAP4 633
002787      000    C                                                                              QAP4 634
002788      000          EQUIVALENCE( IWORK([illegible]),[illegible]001))                         QAP4 635
002789      000    C                                                                              QAP4 636
002790      000    C                                                                              QAP4 637
002791      000    C   UPDATE THE COUNTER FOR THE [illegible]R OF CALLS                           QAP4 638
002792      000          KLBND=KLBND+1                                                            QAP4 639
002793      000    C                                                                              QAP4 640
002794      000          IF(KSNSW(21).EQ.1.AND.MOVE.[illegible]WRITE(6,18)                        QAP4 641
002795      000          IF(KSNSW(21).EQ.1.AND.MOVE.[illegible]WRITE(6,19)ITER,LEVY               QAP4 642
002796      000          IF(KSNSW(21).EQ.1.AND.MOVE.[illegible]WRITE(6,21)ITER,LEVY               QAP4 643
002797      000    C                                                                              QAP4 644
002798      000          IC1=IC1+IWD                                                              QAP4 645
002799      000    C                                                                              QAP4 646
002800      000          IF(KSNSW(22).EQ.1)WRITE(6,22)IC1                                         QAP4 647
002801      000    C                                                                              QAP4 648
002802      000    C   CALCULATE THE IC2 MATRIX                                                  QAP4 649
002803      000      200 IROW=0                                                                   QAP4 650
002804      000          IC2=0                                                                    QAP4 651
002805      000          IF(MOVE.NE.0)GO TO 270                                                   QAP4 652
002806      000          IROW=NFCIL                                                               QAP4 653
002807      000          GO TO 580                                                                QAP4 654
002808      000      270 DO 400 I=1,NFCIL                                                         QAP4 655
002809      000          IF(FCFLG(I))GO TO 400                                                    QAP4 656
002810      000          IROW=IROW+1                                                              QAP4 657
002811      000          WORK2(IROW)=32767                                                        QAP4 658
002812      000          ICOL=0                                                                   QAP4 659
002813      000          DO 350 J=1,NLOCT                                                         QAP4 660
002814      000          IF(FLOGT(J))GO TO 350                                                    QAP4 661
002815      000          ICOL=ICOL+1                                                              QAP4 662
002816      000          IF(.NOT.BANED(I,J))GO TO 280                                             QAP4 663
002817      000          KEEP=32767                                                               QAP4 664
002818      000          GO TO 340                                                                QAP4 665
```

```
002819      000      280 KEEP=0                                                    GAP4 666
002820      000          DO 330 K=1,NFCIL                                          GAP4 667
002821      000          IF(.NOT.FCFLG(K))GO TO 350                                GAP4 668
002822      000          L=LOCAT(K)                                                GAP4 669
002823      000          ISAV=F(I,K)                                               GAP4 670
002824      000          IF(F(K,I).GT.F(I,K))ISAV=F(K,I)                           GAP4 671
002825      000          KEEP=KEEP+(ISAV-D(J,L))                                   GAP4 672
002826      000      330 CONTINUE                                                  GAP4 673
002827      000      340 WORK1(IROW,ICOL)=KEEP                                     GAP4 674
002828      000          IF(KEEP.GE.WORK2(IROW))GO TO 350                          GAP4 675
002829      000          WORK2(IROW)=KEEP                                          GAP4 676
002830      000      350 CONTINUE                                                  GAP4 677
002831      000          IC2=IC2+WORK2(IROW)                                       GAP4 678
002832      000      400 CONTINUE                                                  GAP4 679
002833      000      C                                                             GAP4 680
002834      000      C   WRITE THE IC2 MATRIX IF YOU WISH                         GAP4 681
002835      000          IF(KSNSW(23).EQ.0)GO TO 460                               GAP4 682
002836      000          KEEP=0                                                    GAP4 683
002837      000          DO 430 I=1,NFCIL                                          GAP4 684
002838      000          IF(FCFLG(I))GO TO 430                                     GAP4 685
002839      000          KEEP=KEEP+1                                               GAP4 686
002840      000          INDEX(KEEP)=I                                             GAP4 687
002841      000      430 CONTINUE                                                  GAP4 688
002842      000          KEEP=0                                                    GAP4 689
002843      000          DO 450 J=1,NLOCT                                          GAP4 690
002844      000          IF(FLOGT(J))GO TO 450                                     GAP4 691
002845      000          KEEP=KEEP+1                                               GAP4 692
002846      000          JNDEX(KEEP)=J                                             GAP4 693
002847      000      450 CONTINUE                                                  GAP4 694
002848      000          WRITE(6,23)                                               GAP4 695
002849      000          WRITE(6,11)(JNDEX(J),J=1,KEEP)                            GAP4 696
002850      000          DO 460 I=1,IROW                                           GAP4 697
002851      000      460 WRITE(6,1)INDEX(I),(WORK1(I,J),J=1,ICOL),WORK2(I)         GAP4 698
002852      000          WRITE(6,7)                                                GAP4 699
002853      000      C   CALCULATE THE MODIFIED MATRIX AND GET COLUMN MINIMA       GAP4 700
002854      000      460 DO 500 J=1,ICOL                                           GAP4 701
002855      000          WORK3(J)=32767                                            GAP4 702
002856      000          DO 490 I=1,IROW                                           GAP4 703
002857      000          WORK1(I,J)=WORK1(I,J)-WORK2(I)                            GAP4 704
002858      000          IF(WORK1(I,J).GE.WORK3(J))GO TO 490                       GAP4 705
002859      000          WORK3(J)=WORK1(I,J)                                       GAP4 706
002860      000      490 CONTINUE                                                  GAP4 707
002861      000      500 CONTINUE                                                  GAP4 708
002862      000      C                                                             GAP4 709
002863      000      C   PRINT COLUMN MINIMA                                       GAP4 710
002864      000          IF(KSNSW(23).EQ.0)GO TO 520                               GAP4 711
002865      000          WRITE(6,11)(WORK3(J),J=1,ICOL)                            GAP4 712
002866      000      C                                                             GAP4 713
002867      000      C   IN CASE ICOL.GT.IROW THEN THE FIRST IROW COLUMNS AFTER RANKING GAP4 714
002868      000      520 IF(ICOL.EQ.IROW)GO TO 540                                 GAP4 715
002869      000          CALL ARRNG(ICOL,1,1)                                      GAP4 716
002870      000      540 DO 550 J=1,IROW                                           GAP4 717
002871      000      550 IC2=IC2+WORK3(J)                                          GAP4 718
002872      000      C                                                             GAP4 719
002873      000      C   WRITE RHE VALUES OF IC2 IF YOU WISH                       GAP4 720
002874      000          IF(KSNSW(24).EQ.1)WRITE(6,24)IC2                          GAP4 721
002875      000      C                                                             GAP4 722
```

```
002876      000    C                                                                QAP4 723
002877      000    C   CALCULATE THE VALUE OD IC3                                   QAP4 724
002878      000      580 IC3=0                                                      QAP4 725
002879      000          IF(IROW.EQ.1)GO TO 700                                     QAP4 726
002880      000          IF(KSNSW(26).EQ.1)WRITE(6,56)                              QAP4 727
002881      000          KEEP=0                                                     QAP4 728
002882      000          K=1                                                        QAP4 729
002883      000          IROW2=IROW*(IROW-1)/2                                      QAP4 730
002884      000          DO 650 I=1,N2                                              QAP4 731
002885      000          IF(FLGDS(I))GO TO 650                                      QAP4 732
002886      000          DO 620 J=K,M2                                              QAP4 733
002887      000          IF(FLGIN(J))GO TO 620                                      QAP4 734
002888      000          KEEP=KEEP+1                                                QAP4 735
002889      000          II=DISTI(I)                                                QAP4 736
002890      000          IJ=DISTJ(I)                                                QAP4 737
002891      000          JI=INTRI(J)                                                QAP4 738
002892      000          JJ=INTRJ(J)                                                QAP4 739
002893      000          KAM=D(II,IJ)*F(JI,JJ)                                      QAP4 740
002894      000          IC3=IC3+KAM                                                QAP4 741
002895      000          IF(KSNSW(26).EQ.1)WRITE(6,12)KEEP,II,IJ,JI,JJ,D(II,IJ),F(JI,JJ)  QAP4 742
002896      000         *,KAM                                                       QAP4 743
002897      000          GO TO 630                                                  QAP4 744
002898      000      620 CONTINUE                                                   QAP4 745
002899      000      630 IF(KEEP.EQ.IROW2)GO TO 700                                 QAP4 746
002900      000          K=J+1                                                      QAP4 747
002901      000      650 CONTINUE                                                   QAP4 748
002902      000          DO 660 I=1,10                                              QAP4 749
002903      000      660 WRITE(6,98)KEEP,IROW,IROW2                                 QAP4 750
002904      000          STOP                                                       QAP4 751
002905      000    C                                                                QAP4 752
002906      000      700 IF(KSNSW(27).EQ.1)WRITE(6,14)IC3                           QAP4 753
002907      000          IF(MOVE.NE.0)GO TO 720                                     QAP4 754
002908      000          KEEP=IC1+IC2+IC3                                           QAP4 755
002909      000          IF(KEEP.GT.LBCST)LBCST=KEEP                                QAP4 756
002910      000          WRITE(6,27)KEEP,LBCST                                      QAP4 757
002911      000          GO TO 750                                                  QAP4 758
002912      000      720 LBCST=IC1+IC2+IC3                                          QAP4 759
002913      000      750 IF(KSNSW(28).EQ.1)WRITE(6,16)LBCST                         QAP4 760
002914      000          IF(KSNSW(29).EQ.1)WRITE(6,17)IC1,IC2,IC3,LBCST             QAP4 761
002915      000    C                                                                QAP4 762
002916      000    C                                                                QAP4 763
002917      000        1 FORMAT(10X,20I5)                                           QAP4 764
002918      000        7 FORMAT(/)                                                  QAP4 765
002919      000       11 FORMAT(15X,20I5/)                                          QAP4 766
002920      000       12 FORMAT(12X,I2,6X,I2,2X,I2,5X,I2,3X,I2,3(7X,I4))           QAP4 767
002921      000       14 FORMAT(90X,'VALUE OF C3 =',I5/)                           QAP4 768
002922      000       16 FORMAT(90X,'LOWER BOUND =',I5//)                          QAP4 769
002923      000       17 FORMAT(/50X,'C1=',I5,5X,'C2=',I5,5X,'C3=',I5,10X,'LBCST=',I7)  QAP4 770
002924      000       18 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT THE START')  QAP4 771
002925      000       19 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT ITER ',I3,'  QAP4 772
002926      000         *LEVEL',I3,' ON A FORWARD MOVE')                           QAP4 773
002927      000       21 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT ITER ',I3,'  QAP4 774
002928      000         *LEVEL',I3,' ON A BACKWARD MOVE')                          QAP4 775
002929      000       22 FORMAT(90X,'VALUE IF C1 =',I5/)                           QAP4 776
002930      000       23 FORMAT(10X,'MATRIX FOR CALCULATING C2 FOLLOWS'/)          QAP4 777
002931      000       24 FORMAT(90X,'VALUE OF C2 =',I5/)                           QAP4 778
002932      000       26 FORMAT(10X,'ARRAY OF ELEMENTS TO CALCULATE C3 FOLLOWS'//,  QAP4 779
```

```
002933      000        *      10X,'INDEX    FROM/TO    INTER DET,    DIST,    INTER    VAL QAP4 780
002934      000        *UE')                                                                QAP4 781
002935      000       27 FORMAT(/10X,'CALCULATED STARTING LB=',I5,'  SO WE TAKE LBCST=',    QAP4 782
002936      000        *I5)                                                                 QAP4 783
002937      000       98 FORMAT(02X,20(1H*),3(' ERROR IN LWBND ='),' KEEP=',I3,' IROW=',I3 QAP4 784
002938      000        *',I3,'  IROW2=',I3)                                                 QAP4 785
002939      000         RETURN                                                             QAP4 786
002940      000         END                                                                QAP4 787
002941      000    @FLT,SI TUMER-S*SINAN-QAP4.MAX,,,2355n5053012
002942      000         FUNCTION MAX(K,L)                                                  QAP4 788
002943      000         MAX=K                                                              QAP4 789
002944      000         IF(L.GT.K)MAX=L                                                    QAP4 790
002945      000         RETURN                                                             QAP4 791
002946      000         END                                                                QAP4 792
002947      000    @ELT,SI TUMER-S*SINAN-QAP4,,,
```

```
002990      000          DO 50 I=1,NFCIL                                                    QAP4  835
002991      000          DO 45 J=1,NFCIL                                                    QAP4  836
002992      000          IF(I.GT.J)F(I,J)=0                                                 QAP4  837
002993      000       45 CONTINUE                                                           QAP4  838
002994      000       50 CONTINUE                                                           QAP4  839
002995      000    C                                                                        QAP4  840
002996      000          KSNSW(1)=0                                                         QAP4  841
002997      000          KSNSW(4)=1                                                         QAP4  842
002998      000          KSNSW(6)=1                                                         QAP4  843
002999      000          KSNSW(07)=1                                                        QAP4  844
003000      000          KSNSW(08)=1                                                        QAP4  845
003001      000    C   READ THE INDICES OF THE FACILITIES FIXED APRIORI                    QAP4  846
003002      000          IF(NFIXD.EQ.0)GO TO 80                                             QAP4  847
003003      000          DO 70 I=1,NFIXD                                                    QAP4  848
003004      000       70 READ(5,1)  FIX(I),WHERE(I)                                         QAP4  849
003005      000    C                                                                        QAP4  850
003006      000       80 RETURN                                                             QAP4  851
003007      000    C                                                                        QAP4  852
003008      000    C                                                                        QAP4  853
003009      000    C                                                                        QAP4  854
003010      000        1 FORMAT(20X,12I5)                                                   QAP4  855
003011      000        2 FORMAT(10X,35I2)                                                   QAP4  856
003012      000
003013      000          END
003014      000    @ELT,SI TUMER-S*SINAN-QAP4.TIMOUT,,,205745060512
003015      000          SUBROUTINE TIMOUT
003016      000          COMMON IC1,IC2,IC3,NFCIL,ILOCT,N2,M2,LBCST,ITOT
003017      000          COMMON MSTER(04000),IWORK(2000),KSNSW(60)
003018      000          COMMON KLBND,KUPDT
003019      000          K=ITIME(I,J)
003020      000          T=I/5000.
003021      000          WRITE(6,2000)T
003022      000     2000 FORMAT(' TIME',F10.3,' SEC')
003023      000          RETURN
003024      000          END
003025      000    @ELT,SI TUMER-S*SINAN-QAP7.MAIN,,,130663053112
003026      000          COMMON IC1,IC2,IC3,NFCIL,ILOCT,NFIXD,LBCST,ITOT                    QAP7    1
003027      000          COMMON MSTER(06200),IWORK(2000),KSNSW(60)                          QAP7    2
003028      000          COMMON KLBND,KHADY,KUPDT,ITIME                                     QAP7    3
003029      000          INTEGER*2 F(40,40),D(40,40),DCST(40,40),DISTV(40,40),INTRI(40,40)QAP7    4
003030      000          INTEGER*2 INTVL(40,40),LEVEL(100),LOCAT(40),FCLTY(40),PTCST(40)    QAP7    5
003031      000          INTEGER*2 FIXED(40),WHERE(40),CMCST(40),  FIX(40),WHERE(40)        QAP7    6
003032      000          INTEGER*2 IPASS(40),ICOST(100),ILOCT(100,40),FOUND(100)            QAP7    7
003033      000          INTEGER*2 JBAS(40,40),LINK(40)                                     QAP7    8
003034      000          LOGICAL*1 FCFLG(40),FLOCT(40),FLGDS(40,40),FLGIN(40,40)            QAP7    9
003035      000          LOGICAL*1 BANED(40,40)                                             QAP7   10
003036      000    C                                                                        QAP7   11
003037      000          LOGICAL*1 WORK2(1220),WORK8(1220)                                  QAP7   12
003038      000          INTEGER*2 WORK3(0120),WORK6(0040),WORK7(0120)                      QAP7   13
003039      000    C                                                                        QAP7   14
003040      000          EQUIVALENCE( MSTER(0001),F(1,1))                                   QAP7   11
003041      000          EQUIVALENCE( MSTER(0601),D(1,1))                                   QAP7   16
003042      000    C                                                                        QAP7   17
003043      000          EQUIVALENCE( MSTER(1601),DISTI(1))                                 QAP7   18
003044      000          EQUIVALENCE( MSTER(2901),DISTV(1))                                 QAP7   19
003045      000          EQUIVALENCE( MSTER(3201),INTRI(1))                                 QAP7   20
003046      000          EQUIVALENCE( MSTER(4001),INTVL(1))                                 QAP7   21
```

```
003047  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  22
003048  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  23
003049  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  24
003050  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  25
003051  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  26
003052  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  27
003053  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  28
003054  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  29
003055  000        EQUIVALENCE( MSTR(K(1,1)),KSNSW(1,1))                 QAP7  30
003056  000        EQUIVALENCE( MSTR(K(1,1)),KSNSW(1,1))                 QAP7  31
003057  000        EQUIVALENCE( MSTR(K(1,1)),KSNSW(1,1))                 QAP7  32
003058  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  33
003059  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  34
003060  000  C                                                           QAP7  35
003061  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  36
003062  000        EQUIVALENCE( MSTR(K(1,1)),ASSD(1))                    QAP7  37
003063  000  C                                                           QAP7  38
003064  000        EQUIVALENCE( IWORK(1,1),KSNSW(1,1))                   QAP7  39
003065  000        EQUIVALENCE( IWORK(1,1),KSNSW(1,1))                   QAP7  40
003066  000  C                                                           QAP7  41
003067  000  C                                                           QAP7  42
003068  000  C   READ THE KEYS SETTINGS FOR THE MAIN PROGRAM             QAP7  43
003069  000        READ(5,3)(KSNSW(I),I=1,20)                            QAP7  44
003070  000  C   READ THE KEY SETTINGS FOR SUBROUTINE LWBND2             QAP7  45
003071  000        READ(5,3)(KSNSW(I),I=1,30)                            QAP7  46
003072  000  C                                                           QAP7  47
003073  000  C   READ THE PROBLEM DATA                                   QAP7  48
003074  000     10 CALL DTAIN(JOB,IFMAC,IOPT,NLOCT,NFCIL,ITERM)          QAP7  49
003075  000  C                                                           QAP7  50
003076  000  C   PRINT THE INITIAL DATA IF REQUIRED                      QAP7  51
003077  000        IF(KSNSW(01).EQ.0)GO TO 110                           QAP7  52
003078  000        WRITE(6,4)JOB,NFCIL,NLOCT                             QAP7  53
003079  000        DO 70 I=1,2                                           QAP7  54
003080  000     70 WRITE(6,2)                                            QAP7  55
003081  000        WRITE(6,5)                                            QAP7  56
003082  000        DO 80 I=1,NLOCT                                       QAP7  57
003083  000     80 WRITE(6,01)(D(I,J),J=1,NLOCT)                         QAP7  58
003084  000        WRITE(6,8)                                            QAP7  59
003085  000        DO 90 I=1,NFCIL                                       QAP7  60
003086  000     90 WRITE(6,01)(F(I,J),J=1,NFCIL)                         QAP7  61
003087  000        WRITE(6,07)                                           QAP7  62
003088  000        WRITE(6,5))LBCST,ITOT                                 QAP7  63
003089  000  C   INITIALIZE THE COST                                     QAP7  64
003090  000     110 IC1=0                                                QAP7  65
003091  000        IF(NFIXD.EQ.1)GO TO 125                               QAP7  66
003092  000        DO 120 I=1,10                                         QAP7  67
003093  000     120 WRITE(6,98)NFIXD                                     QAP7  68
003094  000        GO TO 900                                             QAP7  69
003095  000  C   UNFLAG ALL THE LOCATIONS AND THE FACILITIES             QAP7  70
003096  000     125 DO 130 I=1,NLOCT                                     QAP7  71
003097  000     130 FLOGT(I)=.FALSE.                                     QAP7  72
003098  000        DO 140 J=1,NFCIL                                      QAP7  73
003099  000     140 FCFLG(J)=.FALSE.                                     QAP7  74
003100  000  C                                                           QAP7  75
003101  000        DO 150 I=1,NFCIL                                      QAP7  76
003102  000        IPASS(I)=0                                            QAP7  77
003103  000        DO 150 J=1,NLOCT                                      QAP7  78
```

```
003104  000      150 BANED(I,J)=.FALSE.                                          QAP7  79
003105  000    C                                                                 QAP7  80
003106  000    C   INITIALIZE WORK7                                              QAP7  81
003107  000          DO 325 I=1,120                                              QAP7  82
003108  000      325 WORK7(I)=0                                                  QAP7  83
003109  000    C   INITIALIZE WORK8                                              QAP7  84
003110  000          DO 330 I=1,1220                                             QAP7  85
003111  000      330 WORK8(I)=.FALSE.                                            QAP7  86
003112  000          DO 335 I=1,40                                               QAP7  87
003113  000      335 CMCST(I)=0                                                  QAP7  88
003114  000    C                                                                 QAP7  89
003115  000    C                                                                 QAP7  90
003116  000    C                                                                 QAP7  91
003117  000    C                                                                 QAP7  92
003118  000          TIME=0                                                      QAP7  93
003119  000          IEVAL=0                                                     QAP7  94
003120  000          IUBCS=0                                                     QAP7  95
003121  000          KOMPL=0                                                     QAP7  96
003122  000          FRAC=IFRAC                                                  QAP7  97
003123  000          FRAC=0.01*FRAC                                              QAP7  98
003124  000    C                                                                 QAP7  99
003125  000    C   INITIALIZE SUBROUTINE COUNTERS AND FATHOMING COUNTER           QAP7 100
003126  000          KLBND=0                                                     QAP7 101
003127  000          KHADY=0                                                     QAP7 102
003128  000          KCARY=0                                                     QAP7 103
003129  000          KFATH=0                                                     QAP7 104
003130  000          NECTP=0                                                     QAP7 105
003131  000    C                                                                 QAP7 106
003132  000    C   CALCULATE STARTING UPPER AND LOWER BOUNDS                     QAP7 107
003133  000          CALL LWBND2(0,0,0,0,1)                                      QAP7 108
003134  000          LBTC=LBCST                                                  QAP7 109
003135  000          IF(KSNSW(04).EQ.1)WRITE(6,17)LBTC,ITOT                      QAP7 110
003136  000    C                                                                 QAP7 111
003137  000    C     PROCEED TO OBTAIN A SOLUTION ALONG THE RIGHTMOST BRANCH OF A TREE QAP7 112
003138  000          ITER=1                                                      QAP7 113
003139  000          LEVY=1                                                      QAP7 114
003140  000          L2=  FIX(1)                                                 QAP7 115
003141  000           J=WHERE(1)                                                 QAP7 116
003142  000          LEVEL(L2)=LEVY                                              QAP7 117
003143  000          LOCAT(L2)=J                                                 QAP7 118
003144  000          FCFLG(L2)=.TRUE.                                            QAP7 119
003145  000          FLOGT( J)=.TRUE.                                            QAP7 120
003146  000          IF(KSNSW(05).EQ.1)WRITE(6,9)ITER,LEVY,L2,J                  QAP7 121
003147  000          PTCST(1)=0                                                  QAP7 122
003148  000          CMCST(1)=0                                                  QAP7 123
003149  000    C                                                                 QAP7 124
003150  000          J=FIX(1)                                                    QAP7 125
003151  000          FCLTY(1)=J                                                  QAP7 126
003152  000          LEVEL(J)=1                                                  QAP7 127
003153  000          FIXED(1)=J                                                  QAP7 128
003154  000          LIXED(1)=WHERE(1)                                           QAP7 129
003155  000    C   B) CHOOSE THE ELEMENT WITH MAXIMUM INTERACTIONS WITH THE ONE ASSIGNQAP7 130
003156  000    C   AT THE PRECEDING LEVEL                                        QAP7 131
003157  000      430 LEVY=LEVY+1                                                 QAP7 132
003158  000          IPASS(LEVY)=IPASS(LEVY)+1                                   QAP7 133
003159  000          ITER=ITER+1                                                 QAP7 134
003160  000          JET=FCLTY(LEVY-1)                                           QAP7 135
```

```
003161      000          ISAV=-32767                                                    GAP7 136
003162      000          DO 460 I=1,NFCIL                                               GAP7 137
003163      000          IF(FCFLG(I))GO TO 460                                          GAP7 138
003164      000          IF(F(I,JET).LE.ISAV)GO TO 460                                  GAP7 139
003165      000          ISAV=F(I,JET)                                                  GAP7 140
003166      000          KEEP=I                                                         GAP7 141
003167      000      460 CONTINUE                                                       GAP7 142
003168      000      470 FCFLG(KEEP)=.TRUE.                                             GAP7 143
003169      000          LEVEL(KEEP)=LEVY                                               GAP7 144
003170      000      C   C) ASSIGN THE ELEMENT TO A LOCATION SO THAT THE COST OF SUCH AN ASSIGAP7 145
003171      000      C       IS MINIMAL                                                 GAP7 146
003172      000      510 IWD=32767                                                      GAP7 147
003173      000          DO 530 J=1,NLOCT                                               GAP7 148
003174      000          IF(BANED(KEEP,J))GO TO 530                                     GAP7 149
003175      000          IF(FLOGT(J))GO TO 530                                          GAP7 150
003176      000          KAM=0                                                          GAP7 151
003177      000          DO 520 I=1,NFCIL                                               GAP7 152
003178      000          IF(I.LO.KEEP)GO TO 520                                         GAP7 153
003179      000          IF(.NOT.FCFLG(I))GO TO 520                                     GAP7 154
003180      000          L=LOCAT(I)                                                     GAP7 155
003181      000          ISAVV=F(KEEP,I)                                                GAP7 156
003182      000          KAM=KAM+(ISAVV*D(J,L))                                         GAP7 157
003183      000      520 CONTINUE                                                       GAP7 158
003184      000          IF(KAM.GT.IWD)GO TO 530                                        GAP7 159
003185      000          IWD=KAM                                                        GAP7 160
003186      000          LOCAT(KEEP)=J                                                  GAP7 161
003187      000      530 CONTINUE                                                       GAP7 162
003188      000          IF(IWD.EQ.32767)GO TO 730                                      GAP7 163
003189      000          IF(KSNSW(05).EQ.1)WRITE(06,9)ITER,LEVY,KEEP,LOCAT(KEEP),IWD   GAP7 164
003190      000          J=LOCAT(KEEP)                                                  GAP7 165
003191      000          FCLTY(LEVY)=KEEP                                               GAP7 166
003192      000          FLOGT(J)=.TRUE.                                                GAP7 167
003193      000          IUBCS=IUBCS+IWD                                                GAP7 168
003194      000          FIXED(LEVY)=KEEP                                               GAP7 169
003195      000          FTCST(LEVY)=IWD                                                GAP7 170
003196      000          CNCST(LEVY)=IUBCS                                              GAP7 171
003197      000          LIXED(LEVY)=J                                                  GAP7 172
003198      000      C                                                                  GAP7 173
003199      000      C   CHECK WHETHER WE HAVE REACHED A TERMINAL NODE OR NOT           GAP7 174
003200      000      580 IF( LEVY.LT.NFCIL.AND.KOMPL.EQ.0)GO TO 430                     GAP7 175
003201      000          IF( LEVY.EQ.NFCIL)GO TO 590                                    GAP7 176
003202      000      C   CHECK WHETHER IT PAYS TO CONTINUE THE SEARCH ALONG THIS BRANCH GAP7 177
003203      000          CALL LWBND2(LEVY,ITOT,IUBCS,1,P)                               GAP7 178
003204      000          IF(LBCST.LT.ITOT)GO TO 585                                     GAP7 179
003205      000          KPATH=KPATH+1                                                  GAP7 180
003206      000          GO TO 620                                                      GAP7 181
003207      000      585 LEVY=LEVY+1                                                    GAP7 182
003208      000          IPASS(LEVY)=IPASS(LEVY)+1                                      GAP7 183
003209      000          GO TO 730                                                      GAP7 184
003210      000      C   E) NOW WE HAVE A COMPLETE SOLUTION . PRINT IT IF YOU WISH      GAP7 185
003211      000      590 KOMPL=1                                                        GAP7 186
003212      000          IC1=IUBCS                                                      GAP7 187
003213      000          IF(IUBCS.GE.ITOT)GO TO 620                                     GAP7 188
003214      000          ITOT=IUBCS                                                     GAP7 189
003215      000          IEVAL=IEVAL+1                                                  GAP7 190
003216      000          ICOST(IEVAL)=IUBCS                                             GAP7 191
003217      000          FOUND(IEVAL)=ITER                                              GAP7 192
```

```
003216      000          DO 595 I=1,NFCIL                                          GAP7 193
003219      000          ILOCT(IEVAL,I)=LOCAT(I)                                   GAP7 194
003220      000        595 CONTINUE                                                GAP7 195
003221      000        600 IF(KSNSW(06).EQ.0)GO TO 620                             GAP7 196
003224      000          WRITE(6,11)ITER                                           GAP7 197
003225      000          WRITE(6,12)                                               GAP7 198
003226      000          DO 605 I=1,LEVY                                           GAP7 199
003227      000        605 WRITE(6,14)I,FIXED(I),LIXED(I),PTCST(I),CMCST(I)        GAP7 200
003230      000          CALL TIMOUT
003231      000    C                                                               GAP7 201
003232      000    C   REDUCE THE CURRENT UPPER BOUND                             GAP7 202
003233      000        610 DIF=ITOT-LBTC                                           GAP7 203
003234      000          KETOT=ITOT                                                GAP7 204
003235      000          KELBC=LBTC                                                GAP7 205
003236      000          IF(DIF.LE.ISTOP)GO TO 680                                 GAP7 206
003237      000          DIF=(FRAC*DIF)                                            GAP7 207
003240      000          ITOT=LBTC+DIF                                             GAP7 208
003241      000          IF(KSNSW(08).EQ.1)WRITE(6,13)KELBC,ITOT,KETOT             GAP7 209
003242      000    C   STORE THE TREE CONFIGURATION AT THE CURRENT NODE SO THAT WE CAN  GAP7 210
003243      000    C   RETRIEVE IT IF WE FAIL TO OBTAIN A BETTER SOLUTION UNDER THE CURRENTGAP7 211
003244      000    C   CEILING                                                     GAP7 212
003245      000          DO 612 I=1,1220                                           GAP7 213
003246      000          WORK2(I)=.FALSE.                                          GAP7 214
003247      000          IF(WORK8(I))WORK2(I)=.TRUE.                               GAP7 215
003250      000        612 CONTINUE                                                GAP7 216
003251      000          DO 615 I=1,120                                            GAP7 217
003252      000        615 WORK3(I)=WORK7(I)                                       GAP7 218
003253      000          DO 616 I=1,40                                             GAP7 219
003254      000        616 WORK6(I)=CMCST(I)                                       GAP7 220
003255      000          CALL TIMOUT
003256      000    C                                                               GAP7 221
003257      000    C  *** BACKTRACK TO THE NEXT LEVEL UP. THE TREE                 GAP7 222
003260      000    C   A) FREE THE FACILITY AT THAT LEVEL                         GAP7 223
003261      000        620 DO 640 I=1,NFCIL                                        GAP7 224
003262      000          IF(LEVEL(I).EQ.LEVY)GO TO 650                             GAP7 225
003263      000        640 CONTINUE                                                GAP7 226
003264      000    C                                                               GAP7 227
003265      000        650 FCFLG(I)=.FALSE.                                        GAP7 228
003266      000          DO 670 J=1,NLOCT                                          GAP7 229
003267      000          IF(J.EQ.LOCAT(I))GO TO 680                                GAP7 230
003270      000        670 CONTINUE                                                GAP7 231
003271      000        680 FLOGT(J)=.FALSE.                                        GAP7 232
003272      000          IWD=-PTCST(LEVY)                                          GAP7 233
003273      000          IUBCS=IUBCS+IWD                                           GAP7 234
003274      000          BANED(I,J)=.TRUE.                                         GAP7 235
003275      000    C                                                               GAP7 236
003276      000    C                                                               GAP7 237
003277      000    C                                                               GAP7 238
003300      000    C   C) CALCULATE A LB ON ALL SOLUTIONS EXCLUDING THE LAST      GAP7 239
003301      000          CALL LWBND2(LEVY,ITER,IWD,2,2)                            GAP7 240
003302      000    C                                                               GAP7 241
003303      000    C   D) CHECK WHETHER IT PAYS TO RESUME THE SEARCH ALONG THIS BRANCH  GAP7 242
003304      000          IF(LBCST.LT.ITOT )GO TO 730                               GAP7 243
003305      000          KPATH=KPATH+1                                             GAP7 244
003306      000          KEEP=I                                                    GAP7 245
003307      000          GO TO 740                                                 GAP7 246
003310      000    C                                                               GAP7 247
```

```
003275      000   C   RETRIEVE THE INDEX OF THE FACILITY AT THAT LEVEL                    QAP7 248
003276      000     730 KEEP=FIXED(LEVY)                                                  QAP7 249
003277      000         ITER=ITER+1                                                       QAP7 250
003278      000         GO TO 470                                                         QAP7 251
003279      000   C                                                                       QAP7 252
003280      000     740 DO 760 J=1,NLOCT                                                  QAP7 253
003281      000     760 BANED(KEEP,J)=.FALSE.                                           QAP7 254
003282      000         FCFLG(KEEP)=.FALSE.                                             QAP7 255
003283      000         LEVY=LEVY-1                                                     QAP7 256
003284      000         ITER=ITER+1                                                     QAP7 257
003285      000         IF(LEVY.NE.0.AND.ITER.LT.[illegible])GO TO 620                  QAP7 258
003286      000         IF(LEVY.EQ.0.AND.ITER.LT.[illegible])GO TO 775                  QAP7 259
003287      000         WRITE(6,6)                                                      QAP7 260
003288      000         WRITE(6,39)LBTC,ITOT                                            QAP7 261
003289      000         WRITE(6,7)                                                      QAP7 262
003290      000         DO 765 I=1,2                                                    QAP7 263
003291      000     765 WRITE(6,41)ITER                                                 QAP7 264
003292      000         GO TO 810                                                       QAP7 265
003293      000   C   SINCE NO BETTER SOLUTION WAS FOUND , FLIP THE BOUNDS , RESTORE THE  QAP7 266
003294      000   C   ARRAYS AND RETURN TO [illegible]                                  QAP7 267
003295      000     775 IF(KSNSW(08).EQ.1)[illegible]LBTC,ITOT                          QAP7 268
003296      000         IF(KSNSW(08).EQ.1)[illegible]ITER                               QAP7 269
003297      000         CALL TIMOUT                                                     QAP7 270
003298      000         LBTC=ITOT                                                       QAP7 271
003299      000         KETOT=LBTC                                                      QAP7 272
003300      000         IC1=KETOT                                                       QAP7 273
003301      000         IUBCS=KETOT                                                     QAP7 274
003302      000         ITOT=KETOT                                                      QAP7 275
003303      000          DIF=ITOT-LBTC                                                  QAP7 276
003304      000         IF(DIF.LT.ISTOP)GO TO 778                                       QAP7 277
003305      000          DIF=FRAC*DIF                                                   QAP7 278
003306      000         GO TO 780                                                       QAP7 279
003307      000     778 IF(KSNSW(08).EQ.1)WRITE(6,37)DIF                                QAP7 280
003308      000         GO TO 800                                                       QAP7 281
003309      000   C                                                                     QAP7 282
003310      000     780 KETOT=ITOT                                                      QAP7 283
003311      000         ITOT=LBTC+DIF                                                   QAP7 284
003312      000         DO 785 I=1,1220                                                 QAP7 285
003313      000         WORK8(I)=.FALSE.                                                QAP7 286
003314      000         IF(WORK2(I))WORK8(I)=.TRUE.                                     QAP7 287
003315      000     785 CONTINUE                                                        QAP7 288
003316      000         DO 790 I=1,120                                                  QAP7 289
003317      000     790 WORK7(I)=WORK3(I)                                               QAP7 290
003318      000         DO 795 I=1,40                                                   QAP7 291
003319      000     795 CMCST(I)=WORK6(I)                                               QAP7 292
003320      000         LEVY=NFCIL                                                      QAP7 293
003321      000   C                                                                     QAP7 294
003322      000         IF(KSNSW(08).EQ.1)WRITE(6,38)LBTC,ITOT                          QAP7 295
003323      000         GO TO 620                                                       QAP7 296
003324      000   C                                                                     QAP7 297
003325      000   C                                                                     QAP7 298
003326      000   C                                                                     QAP7 299
003327      000     800 WRITE(6,16)                                                     QAP7 300
003328      000     810 KEEP=0                                                          QAP7 301
003329      000         FTHRIO=100.0*FLOAT(KFATH)/FLOAT(KLBND)                          QAP7 302
003330      000         DO 820 I=1,NFCIL                                                QAP7 303
003331      000         LIXED(I)=I                                                      QAP7 304
```

```
003332      000      820 KEEP=KEEP+IPASS(I)                                                    QAP7 303
003333      000          WRITE(6,23)                                                           QAP7 304
003334      000          WRITE(6,24)NFCIL                                                      QAP7 305
003335      000          WRITE(6,26)NLOCT                                                      QAP7 306
003336      000          WRITE(6,07)                                                           QAP7 307
003337      000          WRITE(6,47)NFIXD                                                      QAP7 308
003338      000          WRITE(6,48)                                                           QAP7 309
003339      000          DO 825 I=1,NFIXD                                                      QAP7 310
003340      000      825 WRITE(6,49)I,FIX(I),WHERE(I)                                          QAP7 311
003341      000          WRITE(6,08)                                                           QAP7 312
003342      000          DO 830 I=1,NFCIL                                                      QAP7 313
003343      000      830 WRITE(6,1)(F(I,J),J=1,NFCIL)                                          QAP7 314
003344      000          WRITE(6,05)                                                           QAP7 315
003345      000          DO 840 I=1,NLOCT                                                      QAP7 316
003346      000      840 WRITE(6,1)(D(I,J),J=1,NLOCT)                                          QAP7 317
003347      000          WRITE(6,27)ITER                                                       QAP7 318
003348      000          WRITE(6,28)(IPASS(I),I=1,NFCIL)                                       QAP7 319
003349      000          WRITE(6,29)KEEP                                                       QAP7 320
003350      000          WRITE(6,31)                                                           QAP7 321
003351      000          WRITE(6,32)(LIXED(I),I=1,NFCIL)                                       QAP7 322
003352      000          DO 850 I=1,IEVAL                                                      QAP7 323
003353      000      850 WRITE(6,33)I,ICOST(I),FCOST(I),(TLOCT(I,L),L=1,NFCIL)                 QAP7 324
003354      000          WRITE(6,34)TIME                                                       QAP7 325
003355      000          WRITE(6,07)                                                           QAP7 326
003356      000          WRITE(6,42)                                                           QAP7 327
003357      000          WRITE(6,43)KLBND                                                      QAP7 328
003358      000          WRITE(6,44)NECTP                                                      QAP7 329
003359      000          WRITE(6,52)KHADY                                                      QAP7 330
003360      000          WRITE(6,53)KGARY                                                      QAP7 331
003361      000          WRITE(6,46)FTHRIO                                                     QAP7 332
003362      000          WRITE(6,06)                                                           QAP7 333
003363      000      C                                                                         QAP7 334
003364      000      C                                                                         QAP7 335
003365      000      C   CHECK WHETHER THERE ARE ANY MORE PROBLEMS TO SOLVE                    QAP7 336
003366      000        900 IF(MORE)1000,1000,10                                                QAP7 337
003367      000      C                                                                         QAP7 338
003368      000       1000 STOP                                                              QAP7 339
003369      000      C                                                                         QAP7 340
003370      000          1 FORMAT(10X,40I3)                                                    QAP7 341
003371      000          2 FORMAT(10X,9('*** QAP 7 ***'))                                      QAP7 342
003372      000          3 FORMAT(20X,20I3)                                                    QAP7 343
003373      000          4 FORMAT(1H1,'   DATA FOR PROBLEM',I5/                                QAP7 344
003374      000           *       10X,'      NUMBER OF FACILITIES    =',I3/                  QAP7 345
003375      000           *       10X,'      NUMBER OF LOCATIONS     =',I3//)                QAP7 346
003376      000          5 FORMAT(10X,'    DISTANCE MATRIX'/)                                  QAP7 347
003377      000          6 FORMAT(1H1)                                                         QAP7 348
003378      000          7 FORMAT(/)                                                           QAP7 349
003379      000          8 FORMAT(/10X,'INTERACTION MATRIX'/)                                  QAP7 350
003380      000          9 FORMAT(20X,'AT ITERATION ',I4,'  LEVEL',I3,'  FACILITY',I3,'  IS  QAP7 351
003381      000           *LOCATED',I3,'    WITH AN INCREASE IN COST LEVELS',I5)              QAP7 352
003382      000         11 FORMAT(1H0,5X,'RESULTS OF THE MOST RECENT FEASIBLE SOLUTION , FOUNQAP7 353
003383      000           *D AT ITERATION',I6/)                                               QAP7 354
003384      000         12 FORMAT(20X,'LEVEL     FACILITY   LOCATION   PTCST    CMCST'/)     QAP7 355
003385      000         14 FORMAT(22X,I2,10X,I2,6X,I3,5X,I4,4X,I5)                            QAP7 356
003386      000         16 FORMAT(10X,'END OF CALCULATIONS')                                  QAP7 357
003387      000         17 FORMAT(/10X,'STARTING LOWER BOUND ON THE TOTAL COST=',I5/          QAP7 358
003388      000           *       10X,'STARTING UPPER BOUND ON THE TOTAL COST=',I5/)          QAP7 359
```

```
003389      000        21 FORMAT(10X,'  ARRAY OF [illegible] DISTANCE [illegible]/)                    GAP7 360
003390      000        22 FORMAT(10X,'  ARRAY OF [illegible] INTERAC[illegible] FOLLOWS')             GAP7 361
003391      000        23 FORMAT(1H1,' [illegible] OPTIMAL [illegible] WITH PROBLEM'/)                 GAP7 362
003392      000        24 FORMAT(10X,'NUMBER [illegible] =',I5)                                        GAP7 363
003393      000        26 FORMAT(10X,'NUMBER [illegible] =',I5)                                        GAP7 364
003394      000        27 FORMAT(/10X,'UBOUND [illegible] IS =',I8)                                    GAP7 365
003395      000        28 FORMAT(/10X,'NUMBER [illegible] VARIOUS [illegible]',2UI4)                   GAP7 366
003396      000        29 FORMAT(/10X,'TOTAL [illegible] NODES =',I5)                                  GAP7 367
003397      000        31 FORMAT(/50X,'LOCATIONS OF THE FACILITIES')                                   GAP7 368
003398      000        32 FORMAT( 10X,'NUM ,  [illegible] FOUND AT',2X,2013/)                          GAP7 369
003399      000        33 FORMAT(10X,I3,I7,1[illegible])                                               GAP7 370
003400      000        34 FORMAT(//10X,'TOTAL [illegible] =',F10.2,'  SECONDS')                        GAP7 371
003401      000  C                                                                                     GAP7 372
003402      000        36 FORMAT(/10X,'AS A RESULT OF THIS NEWLY FOUND SOLUTION , WE HAVE ',          GAP7 373
003403      000          *LATEST LB=',I5,'  [illegible] =',I5,'  LB =',I5)                            GAP7 374
003404      000        37 FORMAT(/10X,'HOWEVER , [illegible] NOT SEEM TO BE WORTH IT AS DIF=',        GAP7 375
003405      000          *1PE20.8)                                                                     GAP7 376
003406      000        38 FORMAT(/15X,'THE [illegible] AND UB [illegible] ARE ',2I7)                   GAP7 377
003407      000        39 FORMAT(/10X,'SINCE [illegible] SOLUTION [illegible] ITERATION',I5,' + ',    GAP7 378
003408      000          *I5,'  I WILL RESTART [illegible] LAST COMPLETED ITERATION')                  GAP7 379
003409      000        41 FORMAT(02X,3(' FOUND [illegible] SOLUTION **** [illegible] AFTER ',I8,' ITER GAP7 380
003410      000          *ATIONS')                                                                     GAP7 381
003411      000  C                                                                                     GAP7 382
003412      000        42 FORMAT(10X,'NUMBER OF CASES FOR VARIOUS [illegible]',/)                      GAP7 383
003413      000        43 FORMAT(15X,'SUBROUTINE [illegible] WAS CALLED [illegible]  TIMES')          GAP7 384
003414      000        44 FORMAT(10X,'OUT OF [illegible] WAS NECESSARY TO SOLVE A TP IN',I8,'  GAP7 385
003415      000          *   CASES')                                                                   GAP7 386
003416      000        46 FORMAT(10X,'RATIO [illegible] =',F8.4,'  PER CENT')                          GAP7 387
003417      000  C                                                                                     GAP7 388
003418      000        47 FORMAT(/10X,'NUMBER OF [illegible] ELEMENTS=',I3,'  ARRANGED AS FOLLOWS'GAP7 389
003419      000          */)                                                                           GAP7 390
003420      000        48 FORMAT(10X,'NUM',5X,'[illegible]',8X,'AT LOCATION'/)                         GAP7 391
003421      000        49 FORMAT(10X,I2,9X,I[illegible])                                               GAP7 392
003422      000        51 FORMAT(10X,'START[illegible] FOUND=',I8/                                     GAP7 393
003423      000          *       10X,'START[illegible] [illegible]=',I[illegible])                     GAP7 394
003424      000        52 FORMAT(15X,'SUBROUTINE [illegible] WAS CALLED [illegible]  TIMES')          GAP7 395
003425      000        53 FORMAT(15X,'SUBROUTINE [illegible] WAS CALLED [illegible]  TIMES')          GAP7 396
003426      000        54 FORMAT(/10X,'NO. OF ITERATIONS SO FAR=',I6)                                  GAP7 397
003427      000  C                                                                                     GAP7 398
003428      000        98 FORMAT(02X,4('  **** ERROR ****'),10X,' NFIXD.GT.1, IT EQUALS',            GAP7 399
003429      000          *I5)                                                                          GAP7 400
003430      000  C                                                                                     GAP7 401
003431      000        END                                                                             GAP7 402
003432      000  @ELT,SI TUMER-S*SINAN-GAP7.DATA,,,,13163405511?
003433      000  C   A SUBROUTINE TO READ DATA FOR THE QUADRATIC ASSIGNMENT PROBLEM                    GAP71100
003434      000  C   THIS VERSION IS INTENDED FOR THE NUGENT ET AL DATA                                GAP71101
003435      000        SUBROUTINE DTAIN([illegible],ISTOP,MORE,I[illegible])                           GAP71102
003436      000        COMMON IC1,IC2,IC[illegible],NFIXD,LB[illegible],IOT                            GAP71103
003437      000        COMMON MSTER(06200[illegible]),KSNSW(80)                                        GAP71104
003438      000        COMMON KLRND,KHAD[illegible]                                                    GAP71105
003439      000        INTEGER*2 F(40,40)[illegible] FIX(40),WH[illegible],WORK1(40,40)                GAP71106
003440      000        EQUIVALENCE( MSTER[illegible],F(1,1))                                           GAP71107
003441      000        EQUIVALENCE( MSTER[illegible],D(1,1))                                           GAP71108
003442      000        EQUIVALENCE( MSTE[illegible],  FIX(1))                                          GAP71109
003443      000        EQUIVALENCE( MSTER[illegible],WHERE(1))                                         GAP71110
003444      000        EQUIVALENCE( IWORK(0001),WORK1(5,1))                                            GAP71111
003445      000  C                                                                                     GAP71112
```

```
003446      000    C   READ THE NUMBER OF FACILITIES , THE NUMBER OF LOCATIONS , THE FRACTI GAP71113
003447      000    C   OF THE INTERVAL (ITOT=1000) , THE NUMBER OF FIXED FACILITIES , AND  GAP71114
003448      000    C   THE CONTINUATION PARAMETER                                         GAP71115
003449      000          READ(5,1)JOB,NFCIL,NLOCT,IFRAC,NFIXD,MORE,ISTOP,ITERM,LBCST,ITOT  GAP71116
003450      000    C                                                                       GAP71117
003451      000          IF(ITOT.EQ.0)ITOT=3276700                                        GAP71118
003452      000          DO 10 I=1,40                                                     GAP71119
003453      000          DO 10 J=1,40                                                     GAP71120
003454      000          F(I,J)=0                                                         GAP71121
003455      000       10 D(I,J)=0                                                         GAP71122
003456      000    C                                                                       GAP71123
003457      000          DO 20 I=1,NFCIL                                                  GAP71124
003458      000       20 READ(5,2)(WORK1(I,J),J=1,NFCIL)                                  GAP71125
003459      000    C                                                                       GAP71126
003460      000          DO 30 I=1,NFCIL                                                  GAP71127
003461      000          DO 25 J=1,NFCIL                                                  GAP71128
003462      000          IF(I.EQ.J)GO TO 25                                               GAP71129
003463      000          IF(I.GT.J)F(I,J)=WORK1(I,J)                                      GAP71130
003464      000          IF(I.LT.J)D(I,J)=WORK1(I,J)                                      GAP71131
003465      000       25 CONTINUE                                                         GAP71132
003466      000       30 CONTINUE                                                         GAP71133
003467      000    C                                                                       GAP71134
003468      000          DO 40 I=1,NFCIL                                                  GAP71135
003469      000          DO 40 J=1,NFCIL                                                  GAP71136
003470      000          F(I,J)=F(J,I)                                                    GAP71137
003471      000          D(J,I)=D(I,J)                                                    GAP71138
003472      000       40 CONTINUE                                                         GAP71139
003473      000    C                                                                       GAP71140
003474      000    C                                                                       GAP71141
003475      000          KSNSW(1)=0                                                       GAP71142
003476      000          KSNSW(4)=1                                                       GAP71143
003477      000          KSNSW(6)=1                                                       GAP71144
003478      000          KSNSW(07)=1                                                      GAP71145
003479      000          KSNSW(08)=1                                                      GAP71146
003480      000    C   READ THE INDICES OF THE FACILITIES FIXED APRIORI                  GAP71147
003481      000          DO 70 I=1,NFIXD                                                  GAP71148
003482      000       70 READ(5,1)  FIX(I),WHERE(I)                                       GAP71149
003483      000    C                                                                       GAP71150
003484      000          NFIXD=1                                                          GAP71151
003485      000          RETURN                                                           GAP71152
003486      000    C                                                                       GAP71153
003487      000    C                                                                       GAP71154
003488      000    C                                                                       GAP71155
003489      000        1 FORMAT(20X,12I5)                                                GAP71156
003490      000        2 FORMAT(10X,35I2)                                                GAP71157
003491      000
003492      000          END
003493      000    @FLT,SI TUMER-S*SINAN-GAP7,GARY,,,?06344053012
003494      000    C                                                                       GAP7 899
003495      000    C                                                                       GAP7 903
003496      000    C                                                                       GAP7 904
003497      000    C   ********************************************************************GAP7 905
003498      000    C   A SUBROUTINE TO IDENTIFY BASIS LOOP AND UPDATE ITS ENTRIES          GAP7 906
003499      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++GAP7 907
003500      000          SUBROUTINE GARY(NORIG,KI,NROW1,NCOL1,KEY)                        GAP7 908
003501      000    C   ---------------------------------------------------------------------GAP7 909
003502      000          COMMON IC1,IC2,IC3,NFCIL,NDEST,NFIXD,LBCST,KTOT                  GAP7 910
```

```
003503      000          COMMON MSTER(062001,[illegible]),KSNSW(30)                    GAP7 911
003504      000          COMMON KLOND,KHAD([illegible]                                 GAP7 912
003505      000    C                                                                   GAP7 913
003506      000          INTEGER*2 JBAS([illegible]                                    GAP7 914
003507      000          INTEGER*2 INE[illegible]NFT2(50)                              GAP7 915
003508      000    C                                                                   GAP7 916
003509      000          EQUIVALENCE( IWO[illegible](001))                             GAP7 917
003510      000          EQUIVALENCE( IWOR[illegible](001))                            GAP7 918
003511      000          EQUIVALENCE( IWORK([illegible](001))                          GAP7 919
003512      000          EQUIVALENCE( IWORK(1001), JBAS(5,1))                          GAP7 920
003513      000    C                                                                   GAP7 921
003514      000    C                                                                   GAP7 922
003515      000    C   <<+++++++++++++++++[illegible]++++++++++++++++++++++++++++++++GAP7 923
003516      000    C                                                                   GAP7 924
003517      000    C                                                                   GAP7 925
003518      000    C   UPDATE CALLS COUNTER                                           GAP7 926
003519      000          KGARY=KGARY+1                                                 GAP7 927
003520      000    C                                                                   GAP7 928
003521      000    C   A) INITIALZE                                                    GAP7 929
003522      000    C    * SET K2 .GE. (10**K)                                          GAP7 930
003523      000          K2=300000                                                     GAP7 931
003524      000          K=NORIG+NDEST                                                 GAP7 932
003525      000          K=2*K                                                         GAP7 933
003526      000          DO 20 I=1,K                                                   GAP7 934
003527      000       20 INET(I)=0                                                     GAP7 935
003528      000    C    ** THUS THE SIZE OF ARRAY INET .EQ. 2*(NORIG+NDEST) **        GAP7 936
003529      000          DO 40 I=1,NORIG                                               GAP7 937
003530      000       40 INET1(I)=0                                                    GAP7 938
003531      000          DO 60 I=1,NDEST                                               GAP7 939
003532      000       60 INET2(I)=0                                                    GAP7 940
003533      000    C   <<+++++++++++++++++[illegible]++++++++++++++++++++++++++++++++GAP7 941
003534      000    C   B) START SEARCHING FOR [illegible]S LOOP                       GAP7 942
003535      000    C   ------------------------------------------------------------GAP7 943
003536      000    C      ** FROM HERE UNTIL [illegible] 900 OUR MAIN TASKS ARE       GAP7 944
003537      000    C         1) RECOGNIZING [illegible] OF LOOP AND PUTTING ITS ROW AND GAP7 945
003538      000    C            COLUMN [illegible] IN THE ADJACENT LOCATIONS OF ARRAY INET GAP7 946
003539      000    C         2) DETECT ANY [illegible] ERRORS                         GAP7 947
003540      000    C   ------------------------------------------------------------GAP7 948
003541      000          I=1                                                           GAP7 949
003542      000          INET(I)=NROW1                                                 GAP7 950
003543      000          I1=I+1                                                        GAP7 951
003544      000          INET(I1)=NCOL1                                                GAP7 952
003545      000          NROW=NROW1                                                    GAP7 953
003546      000          NCOL=1                                                        GAP7 954
003547      000          I=I+2                                                         GAP7 955
003548      000      100 IF(JBAS(NROW,NCOL)-K1)160,120,120                           GAP7 956
003549      000      120 IF(NCOL-NCOL1)140,160,[illegible]                           GAP7 957
003550      000      140 INET(I)=NROW                                                GAP7 958
003551      000          I1=I+1                                                        GAP7 959
003552      000          INET(I1)=NCOL                                                 GAP7 960
003553      000          I=I+2                                                         GAP7 961
003554      000          GO TO 220                                                     GAP7 962
003555      000    C                                                                   GAP7 963
003556      000      160 NCOL=NCOL+1                                                 GAP7 964
003557      000          IF(NCOL-NDEST)100,100,[illegible]                           GAP7 965
003558      000      200 I=1                                                         GAP7 966
003559      000    C   *) NO BASIS ELEMENT IN THE ROW OF THE ENTERING , ERROR TYPE 1 GAP7 967
```

```
003560      000          WRITE(6,1)I,NROW1,NCOL1                                    QAP7 968
003561      000          KEY=2                                                      QAP7 969
003562      000    C                                                                QAP7 970
003563      000          RETURN                                                     QAP7 971
003564      000    C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 972
003565      000      240 INET2(NCOL)=1                                              QAP7 973
003566      000          NROW=1                                                     QAP7 974
003567      000      240 IF(JBAS(NROW,NCOL)-K1)260,300,300                          QAP7 975
003568      000      260 NROW=NROW+1                                                QAP7 976
003569      000          IF(NROW-NORIG)240,240,300                                  QAP7 977
003570      000      300 I=I-2                                                      QAP7 978
003571      000          IF(I)360,360,340                                           QAP7 979
003572      000      340 NROW=INET(I)                                               QAP7 980
003573      000          I1=I+1                                                     QAP7 981
003574      000          NCOL=INET(I1)                                              QAP7 982
003575      000          INET2(NCOL)=0                                              QAP7 983
003576      000    C                                                                QAP7 984
003577      000          GO TO 520                                                  QAP7 985
003578      000    C   *) THERE IS NO BASIS LOOP                                    QAP7 986
003579      000      360 I=2                                                        QAP7 987
003580      000    C                                                                QAP7 988
003581      000          WRITE(6,2)I,NROW1,NCOL1                                    QAP7 989
003582      000          KEY=2                                                      QAP7 990
003583      000    C                                                                QAP7 991
003584      000          RETURN                                                     QAP7 992
003585      000    C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 993
003586      000      380 I2=I-2                                                     QAP7 994
003587      000          IF(NROW-INET(I2))400,[illegible]                           QAP7 995
003588      000      400 IF(INET1(NROW))300,420,[illegible]                         QAP7 996
003589      000      420 INET(I)=NROW                                               QAP7 997
003590      000          I1=I+1                                                     QAP7 998
003591      000          INET(I1)=NCOL                                              QAP7 999
003592      000          I=I+2                                                      QAP71000
003593      000          IF(NROW-NROW1)480,460,480                                  QAP71001
003594      000      460 I=I-2                                                      QAP71002
003595      000          GO TO 300                                                  QAP71003
003596      000      480 INET1(NROW)=1                                              QAP71004
003597      000          NCOL=1                                                     QAP71005
003598      000      500 IF(JBAS(NROW,NCOL)-K1)520,560,560                          QAP71006
003599      000      520 NCOL=NCOL+1                                                QAP71007
003600      000          IF(NCOL-NDEST)500,500,640                                  QAP71008
003601      000      560 I1=I-1                                                     QAP71009
003602      000          IF(NCOL-INET(I1))580,520,580                               QAP71010
003603      000      580 IF(INET2(NCOL))640,600,640                                 QAP71011
003604      000      600 INET(I)=NROW                                               QAP71012
003605      000          I1=I+1                                                     QAP71013
003606      000          INET(I1)=NCOL                                              QAP71014
003607      000          I=I+2                                                      QAP71015
003608      000          IF(NCOL-NCOL1)220,700,220                                  QAP71016
003609      000      640 I=I-2                                                      QAP71017
003610      000          IF(I)360,360,680                                           QAP71018
003611      000      680 NROW=INET(I)                                               QAP71019
003612      000          I1=I+1                                                     QAP71020
003613      000          NCOL=INET(I1)                                              QAP71021
003614      000          INET1(NROW)=0                                              QAP71022
003615      000          GO TO 260                                                  QAP71023
003616      000    C                                                                QAP71024
```

```
003617      000      700 I=3                                                        QAP71025
003618      000    C   *) CALCULATE THE MINIMUM ENTRY IN BASIS LOOP                 QAP71026
003619      000        ISAV=K2                                                      QAP71027
003620      000      720 NROW=INET(I)                                               QAP71028
003621      000        I1=I+1                                                       QAP71029
003622      000        NCOL=INET(I1)                                                QAP71030
003623      000        IF(JBAS(NROW,NCOL)-ISAV)740,740,780                          QAP71031
003624      000      740 ISAV=JBAS(NROW,NCOL)                                       QAP71032
003625      000        NROW2=NROW                                                   QAP71033
003626      000        NCOL2=NCOL                                                   QAP71034
003627      000      780 IF(NCOL-NCOL1)800,860,800                                  QAP71035
003628      000      800 I=I+4                                                      QAP71036
003629      000        IF(I-K)720,720,840                                           QAP71037
003630      000    C   *) THE BASIS LOOP HAS MORE ENTRIES THAN THERE ARE IN THE BASIS ,  QAP71038
003631      000    C    ERROR TYPE 3                                                QAP71039
003632      000      840 I=3                                                        QAP71040
003633      000    C                                                                QAP71041
003634      000        WRITE(6,3)I,NROW1,NCOL1                                      QAP71042
003635      000        KEY=2                                                        QAP71043
003636      000    C                                                                QAP71044
003637      000        RETURN                                                       QAP71045
003638      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP71046
003639      000      860 IF(ISAV-K2)900,880,840                                     QAP71047
003640      000    C   *) THERE IS NO ELEMENT IN THE BASIS LOOP .LT. K2 ERROR TYPE 4  QAP71048
003641      000      880 I=4                                                        QAP71049
003642      000    C                                                                QAP71050
003643      000        WRITE(6,4)I,NROW1,NCOL1                                      QAP71051
003644      000        KEY=2                                                        QAP71052
003645      000    C                                                                QAP71053
003646      000        RETURN                                                       QAP71054
003647      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP71055
003648      000    C   C) NOW THE LOCATIONS OF ALL ENTRIES IN THE BASIS ARE SEQUENTIALLY  QAP71056
003649      000    C      ARRANGED (PAIRWISE) IN ORDER OF I AND ISAV CONTAINS THE MINIMUM  QAP71057
003650      000    C      ENTRY IN THE LOOP , SO WE CAN UPDATE THE SIMPLEX SOLUTION BY  QAP71058
003651      000    C      SUCCESSIVE ADDITIONS AND SUBTRACTIONS FROM THE ELEMENTS OF THE  QAP71059
003652      000    C      BASIS LOOP                                                  QAP71060
003653      000    C   ---------------------------------------------------------------QAP71061
003654      000      900 J=-1                                                       QAP71062
003655      000        NROW=INET(1)                                                 QAP71063
003656      000        NCOL=INET(2)                                                 QAP71064
003657      000        JBAS(NROW,NCOL)=ISAV                                         QAP71065
003658      000        ISAV=ISAV-K1                                                 QAP71066
003659      000    C                                                                QAP71067
003660      000        I=3                                                          QAP71068
003661      000      920 NROW=INET(I)                                               QAP71069
003662      000        I1=I+1                                                       QAP71070
003663      000        NCOL=INET(I1)                                                QAP71071
003664      000        IF(NROW-NROW2)1000,940,1000                                  QAP71072
003665      000      940 IF(NCOL-NCOL2)1000,960,1000                                QAP71073
003666      000      960 JBAS(NROW,NCOL)=0                                          QAP71074
003667      000        GO TO 1020                                                   QAP71075
003668      000    C                                                                QAP71076
003669      000     1000 ISET=JBAS(NROW,NCOL)+J*ISAV                                QAP71077
003670      000        JBAS(NROW,NCOL)=ISET                                         QAP71078
003671      000    C                                                                QAP71079
003672      000     1020 J=-J                                                       QAP71080
003673      000        I=I+2                                                        QAP71081
```

```
003674      000          IF(NCOL-NCOL1)920,1000,924                                          QAP71082
003675      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP71083
003676      000    C   NOW THAT THE NEW BASIC SOLUTION HAS BEEN OBTAINED , SET KEY = 1 TO   QAP71084
003677      000    C   ENABLE HADY TO CHECK OPTIMALITY                                      QAP71085
003678      000    C   ------------------------------------------------------------------QAP71086
003679      000     1000 KEY=1                                                              QAP71087
003680      000          RETURN                                                             QAP71088
003681      000    C                                                                        QAP71089
003682      000          1 FORMAT(10X,'ERROR TYPE 1'/10X,'NO BASIS ELEMENT IN ROW WHICH CONTAQAP71090
003683      000           *INS ENTERING ELEMENT IN AN ITERATION ',3I3)                     QAP71091
003684      000          2 FORMAT(10X,'ERROR TYPE 2'/10X,'THERE IS NO DATA LOOP',3I3)      QAP71092
003685      000          3 FORMAT(10X,'ERROR TYPE 3'/10X,'THE BASIS LOOP HAS MORE ELEMENTS THQAP71093
003686      000           *AN THERE ARE IN THE DATA',3I3)                                 QAP71094
003687      000          4 FORMAT(10X,'ERROR TYPE 4'/10X,'THERE IS NO ELEMENT IN THE BASIS QAP71095
003688      000           *OOP LESS THSN K2',3I3)                                         QAP71096
003689      000    C                                                                        QAP71097
003690      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP71098
003691      000          END                                                                QAP71099
003692      000    @ELT,SI TUMER-S*SINAN-QAP7.HADY,...717053012
003693      000    C   ***********************************************************QAP7 718
003694      000    C   A SUBROUTINE TO FIND AN INITIAL SOLUTION OF A TRANSPORTATION PROBLEQAP7 719
003695      000    C   USING THE METHOD OF ROW MINIMA                                     QAP7 720
003696      000    C   APPLICABLE TO DEGENERATE CASES BUT NOT TO UNBALANCED ONES          QAP7 721
003697      000    C   ALL X'S HAVE TO BE UPDATED BY EXTRACTING K1                        QAP7 722
003698      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 723
003699      000    C                                                                        QAP7 724
003700      000          SUBROUTINE HADY(NORIG,K1,NDST1,NCOL1,ITOT,KEY)                     QAP7 725
003701      000    C   ------------------------------------------------------------------QAP7 726
003702      000          COMMON IC1,IC2,IC3,NDEST,NORIG,NFIXD,LBCST,KTOT                    QAP7 727
003703      000          COMMON MSTER(06200),IWORK(6000),KSNSW(80)                          QAP7 728
003704      000          COMMON KLBND,KHADY,KRAY,KTOT                                       QAP7 729
003705      000    C                                                                        QAP7 730
003706      000          INTEGER*2 ICOST(40,40),IOBJC(40),IDEST(40),JBAS(40,40)             QAP7 731
003707      000          DIMENSION IU(40),IV(40),ID(40),IV1(40)                             QAP7 732
003708      000    C                                                                        QAP7 733
003709      000    C                                                                        QAP7 734
003710      000          EQUIVALENCE( IWORK(0001),ICOST(5,1))                               QAP7 735
003711      000          EQUIVALENCE( IWORK(0401),IOBJC(001))                               QAP7 736
003712      000          EQUIVALENCE( IWORK(0421),IDEST(001))                               QAP7 737
003713      000          EQUIVALENCE( IWORK(0461),   IU(001))                               QAP7 738
003714      000          EQUIVALENCE( IWORK(0501),   IV(001))                               QAP7 739
003715      000          EQUIVALENCE( IWORK(0521),  ID1(001))                               QAP7 740
003716      000          EQUIVALENCE( IWORK(0561),  IV1(001))                               QAP7 741
003717      000          EQUIVALENCE( IWORK(1601), JBAS(5,1))                               QAP7 742
003718      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 743
003719      000    C   UPDATE CALLS COUNTER                                               QAP7 744
003720      000          KHADY=KHADY+1                                                      QAP7 745
003721      000    C                                                                        QAP7 746
003722      000          GO TO(320,5),KEY                                                   QAP7 747
003723      000    C   ------------------------------------------------------------------QAP7 748
003724      000    C   INITIALIZE                                                         QAP7 749
003725      000          5 K1=30000                                                         QAP7 750
003726      000            NDST1=NDEST                                                      QAP7 751
003727      000            NORG1=NORIG                                                      QAP7 752
003728      000            NCOL=1                                                           QAP7 753
003729      000            NROW=1                                                           QAP7 754
003730      000            ISAV=K1                                                          QAP7 755
```

```
003731      000          DO 130 I=1,NORIG                                                  QAP7 756
003732      000          DO 130 J=1,NDEST                                                  QAP7 757
003733      000        130 JBAS(I,J)=0                                                     QAP7 758
003734      000          DO 140 J=1,NDEST                                                  QAP7 759
003735      000        140 IV(J)=1                                                         QAP7 760
003736      000      C   FROM HERE UNTIL JUST B[illegible] VARIABLES ARE ALLOCATED        QAP7 761
003737      000        150 IF(ICOST(NROW,NCOL)-ISAV)[illegible]                          QAP7 762
003738      000        160 NCOL=NCOL+1                                                   QAP7 763
003739      000            IF(NCOL-NDEST)150,150,[illegible]                             QAP7 764
003740      000        180 IF(IV(NCOL)-1)160,190,180                                     QAP7 765
003741      000        190 ISAV=ICOST(NROW,NCOL)                                         QAP7 766
003742      000            NCOL1=NCOL                                                    QAP7 767
003743      000            GO TO 160                                                     QAP7 768
003744      000      C                                                                   QAP7 769
003745      000        200 IF(IORIG(NROW)-IDLST(NCOL1))210,230,310                       QAP7 770
003746      000        210 ISET=IORIG(NROW)+K1                                           QAP7 771
003747      000      C                                                                   QAP7 772
003748      000            JBAS(NROW,NCOL1)=ISET                                         QAP7 773
003749      000      C                                                                   QAP7 774
003750      000            IDEST(NCOL1)=IDEST(NCOL1)-IORIG(NROW)                         QAP7 775
003751      000            GO TO 290                                                     QAP7 776
003752      000      C                                                                   QAP7 777
003753      000        230 ISET=IORIG(NROW)+K1                                           QAP7 778
003754      000      C                                                                   QAP7 779
003755      000            JBAS(NROW,NCOL1)=ISET                                         QAP7 780
003756      000      C                                                                   QAP7 781
003757      000            IV(NCOL1)=0                                                   QAP7 782
003758      000      C   MAKE NECESSARY ARRANGEMENTS FOR DEGENERACY                      QAP7 783
003759      000            NCOL=1                                                        QAP7 784
003760      000            ISAV=K1                                                       QAP7 785
003761      000            IF(NROW-NORIG)240,300,[illegible]                             QAP7 786
003762      000        240 IF(ICOST(NROW,NCOL)-[illegible]                               QAP7 787
003763      000        250 IF(IV(NCOL)-1)270,260,[illegible]                             QAP7 788
003764      000        260 ISAV=ICOST(NROW,NCOL)                                         QAP7 789
003765      000            NCOL1=NCOL                                                    QAP7 790
003766      000        270 NCOL=NCOL+1                                                   QAP7 791
003767      000            IF(NCOL-NDEST)240,240,280                                     QAP7 792
003768      000      C                                                                   QAP7 793
003769      000        280 JBAS(NROW,NCOL1)=K1                                           QAP7 794
003770      000      C                                                                   QAP7 795
003771      000        290 NROW=NROW+1                                                   QAP7 796
003772      000            IF(NROW-NORIG)300,300,320                                     QAP7 797
003773      000        300 NCOL=1                                                        QAP7 798
003774      000            ISAV=K1                                                       QAP7 799
003775      000            GO TO 150                                                     QAP7 800
003776      000      C                                                                   QAP7 801
003777      000        310 ISET=IDEST(NCOL1)+K1                                          QAP7 802
003778      000      C                                                                   QAP7 803
003779      000            JBAS(NROW,NCOL1)=ISET                                         QAP7 804
003780      000      C                                                                   QAP7 805
003781      000            IORIG(NROW)=IORIG(NROW)-IDEST(NCOL1)                          QAP7 806
003782      000            IV(NCOL1)=0                                                   QAP7 807
003783      000            GO TO 300                                                     QAP7 808
003784      000      C   <<+++++++++++++++++++++[illegible]                              QAP7 809
003785      000      C   START CALCULATING THE DUAL VARIABLES                            QAP7 810
003786      000      C   ------------------------------------------------------------    QAP7 811
003787      000      C   A) INITIALIZE AS FOLLOWS                                        QAP7 812
```

```
003780      000    C       * SET  ALL DUAL VARIABLES  IU + IV EQUAL TO ZERO                    GAP7 813
003789      000    C       * FALG ALL DUAL VARIABLES, IU1+ IV1 ARE SET EQUAL TO ZERO          GAP7 814
003790      000      320 DO 330 I=1,NORIG                                                     GAP7 815
003791      000          IU(I)=0                                                              GAP7 816
003792      000      330 IU1(I)=0                                                             GAP7 817
003793      000          DO 340 J=1,NDEST                                                     GAP7 818
003794      000          IV(J)=0                                                              GAP7 819
003795      000      340 IV1(J)=0                                                             GAP7 820
003796      000    C   UNFLAG THE FIRST ROW                                                   GAP7 821
003797      000          IU1(1)=1                                                             GAP7 822
003798      000          NROW=1                                                               GAP7 823
003799      000          NCOL=1                                                               GAP7 824
003800      000    C   B) FROM HERE UNTIL JUST BEFORE 540 THE REMAINING DUAL VARIABLES ARE    GAP7 825
003801      000    C       DETERMINED BY RECURSIVE SCANNING                                 GAP7 826
003802      000      360 IF(JBAS(NROW,NCOL)-K1)370,370,370                                  GAP7 827
003803      000      370 IV(NCOL)=ICOST(NROW,NCOL)-IU(NROW)                                 GAP7 828
003804      000    C   * UNFLAG COLUMN NCOL                                                  GAP7 829
003805      000          IV1(NCOL)=1                                                          GAP7 830
003806      000      380 NCOL=NCOL+1                                                        GAP7 831
003807      000          IF(NCOL-NDEST)360,360,400                                          GAP7 832
003808      000      400 NROW=NROW+1                                                        GAP7 833
003809      000          IF(NROW-NORIG)420,420,500                                          GAP7 834
003810      000      420 IF(IU1(NROW)-1)430,[illegible]                                     GAP7 835
003811      000      430 NCOL=1                                                             GAP7 836
003812      000      440 IF(JBAS(NROW,NCOL)-K1)[illegible]                                  GAP7 837
003813      000      450 NCOL=NCOL+1                                                        GAP7 838
003814      000          IF(NCOL-NDEST)440,[illegible]                                      GAP7 839
003815      000      470 IF(IV1(NCOL))480,[illegible]                                       GAP7 840
003816      000      480 IU(NROW)=ICOST(NROW,NCOL)-IV(NCOL)                                 GAP7 841
003817      000    C   * UNFLAG ROW NROW                                                    GAP7 842
003818      000          IU1(NROW)=1                                                          GAP7 843
003819      000          NCOL=1                                                               GAP7 844
003820      000          GO TO 360                                                            GAP7 845
003821      000    C   * CHECK WHETHER ANY ROWS ARE STILL FLAGGED , IF SO RETURN TO 430 ,     GAP7 846
003822      000    C       OTHERWISE GO TO 540                                              GAP7 847
003823      000      500 NROW=1                                                             GAP7 848
003824      000      510 IF(IU1(NROW))520,[illegible]                                       GAP7 849
003825      000      520 NROW=NROW+1                                                        GAP7 850
003826      000          IF(NROW-NORIG)510,[illegible]                                      GAP7 851
003827      000    C   <<+++++++++++++++++++[illegible]++++++++++++++++++++++++++++++++++   GAP7 852
003828      000    C   CHECK OPTIMALITY OF THE [illegible] SOLUTION                         GAP7 853
003829      000    C   ----------[illegible]----------                                       GAP7 854
003830      000      540 ISAV=0                                                             GAP7 855
003831      000          NROW=1                                                             GAP7 856
003832      000      550 NCOL=1                                                             GAP7 857
003833      000      560 IF(JBAS(NROW,NCOL)-K1)[illegible]                                  GAP7 858
003834      000      570 NCOL=NCOL+1                                                        GAP7 859
003835      000          IF(NCOL-NDEST)560,560,[illegible]                                  GAP7 860
003836      000      590 NROW=NROW+1                                                        GAP7 861
003837      000          IF(NROW-NORIG)550,550,610                                          GAP7 862
003838      000      610 IF(ISAV)700,800,700                                                GAP7 863
003839      000      620 IS1=IU(NROW)+IV(NCOL)-ICOST(NROW,NCOL)                             GAP7 864
003840      000          IF(IS1-ISAV)570,570,640                                            GAP7 865
003841      000      640 ISAV=IS1                                                           GAP7 866
003842      000    C                                                                        GAP7 867
003843      000          NROW1=NROW                                                         GAP7 868
003844      000          NCOL1=NCOL                                                         GAP7 869
```

```
003845      000          GO TO 570                                                    GAP7 870
003846      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++>>GAP7 871
003847      000    C   CURRENT SOLUTION IS NON-OPTIMAL                               GAP7 872
003848      000    C   ----------------------------------------------------------    GAP7 873
003849      000      700 KEY=1                                                        GAP7 874
003850      000          RETURN                                                       GAP7 875
003851      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++>>GAP7 876
003852      000    C   CURRENT SOLUTION IS OPTIMAL                                   GAP7 877
003853      000    C   ----------------------------------------------------------    GAP7 878
003854      000      800 KEY=2                                                        GAP7 879
003855      000    C   CALCULATE ACTUAL VALUES OF BASIC VARIABLES                    GAP7 880
003856      000          ITOT=0                                                       GAP7 881
003857      000          DO 850 I=1,NORIG                                             GAP7 882
003858      000          DO 830 J=1,NDEST                                             GAP7 883
003859      000          IF(JBAS(I,J))830,830,820                                     GAP7 884
003860      000      820 JBASE    =JBAS(I,J)                                          GAP7 885
003861      000          ITOT=ITOT+(JBASE     *                                       GAP7 886
003862      000      830 CONTINUE                                                     GAP7 887
003863      000      850 CONTINUE                                                     GAP7 888
003864      000          IF(KSNSW(30).EQ.0)GO TO 900                                 GAP7 889
003865      000          WRITE(6,11)                                                 GAP7 890
003866      000          DO 870 I=1,NORIG                                            GAP7 891
003867      000      870 WRITE(6,01)I,(JBAS(I,J),J=1,NDEST)                          GAP7 892
003868      000    C                                                                 GAP7 893
003869      000      900 RETURN                                                      GAP7 894
003870      000    C                                                                 GAP7 895
003871      000        1 FORMAT(10X,I3,2X,401I)                                      GAP7 896
003872      000       11 FORMAT(/10X,'OPTIMAL ASSIGNMENT OF THE LINEAR ASSIGNMENT PROBLEM '/)GAP7 897
003873      000          END
003874      000    @ELT,SI TUMER-S*SINAH-GAP7.S                                     0247605501
003875      000    C   A SUBROUTINE TO CALCULATE ALL BASIC SOLUTIONS OF ANY MODEL    GAP7 405
003876      000          SUBROUTINE LWBND2(LWV,LWC,LWB,MOVE,KAT)                     GAP7 406
003877      000          COMMON IC1,IC2,IC3,IC4,IC5,NCIXO,LEC,LEB(40)                GAP7 407
003878      000          COMMON MSTER(06200),F(40,40),KSNSW(50)                      GAP7 408
003879      000          COMMON KLRND,KHADY,KOUNT,KLECTP                             GAP7 409
003880      000    C                                                                 GAP7 410
003881      000          INTEGER*2 WORK1(40,40),WORK2(40,40),WORK3(40),WORK4(40),WORK6(40) GAP7 411
003882      000          INTEGER*2 WORK8(40)                                         GAP7 412
003883      000          INTEGER*2 F(40,40),D(40,40),CC(40,40),U(40,40),INTR1(40,40)GAP7 413
003884      000          INTEGER*2 INTVL(40),ROW(40),JBAS(40,40),IDEST(40)           GAP7 414
003885      000          LOGICAL*1 ASIGN(40,40),FLGIN(40,40),FLGT(40)                GAP7 415
003886      000          LOGICAL*1 FCFLG(40),BASIC(40,40)                            GAP7 416
003887      000          EQUIVALENCE( IWORK(0001),WORK1(1,1))                        GAP7 417
003888      000          EQUIVALENCE( IWORK(0401),WORK2(1,1))                        GAP7 418
003889      000          EQUIVALENCE( IWORK(0801),WORK3(001))                        GAP7 419
003890      000          EQUIVALENCE( IWORK(0841),WORK4(001))                        GAP7 420
003891      000          EQUIVALENCE( IWORK(1001),WORK6(1))                          GAP7 421
003892      000          EQUIVALENCE( IWORK(1041),WORK8(  1))                        GAP7 422
003893      000          EQUIVALENCE( IWORK(1081),WORK9(  1))                        GAP7 423
003894      000          EQUIVALENCE( IWORK(1121),WORK5(  1))                        GAP7 424
003895      000          EQUIVALENCE( IWORK(1161),WORK7(  1))                        GAP7 425
003896      000    C                                                                 GAP7 426
003897      000          EQUIVALENCE( MSTER(0001),F(1,1))                            GAP7 427
003898      000          EQUIVALENCE( MSTER(1601),D(1,1))                            GAP7 428
003899      000          EQUIVALENCE( MSTER(3201),DIST(1))                           GAP7 429
003900      000          EQUIVALENCE( MSTER(2401),DEMV(1))                           GAP7 430
003901      000          EQUIVALENCE( MSTER(3201),INTPI(1))                          GAP7 431
```

```
003904      000          EQUIVALENCE( MSTER(6081),NOVL(1))                                  QAP7 432
003905      000          EQUIVALENCE( MSTER(6[illegible]),LOCA(1))                          QAP7 433
003906      000          EQUIVALENCE( MSTER(6[illegible]),IFLG(1))                          QAP7 434
003907      000          EQUIVALENCE( MSTER(6[illegible]),[illegible](1))                   QAP7 435
003908      000          EQUIVALENCE( MSTER(6[illegible]),[illegible](1,1))                 QAP7 436
003907      000          EQUIVALENCE( MSTER(6[illegible]),[illegible](1,1))                 QAP7 437
003908      000          EQUIVALENCE( MSTER(6721),[illegible](1,1))                         QAP7 438
003909      000    C                                                                        QAP7 439
003910      000    C   IPDATE CALLS COUNTERS                                               QAP7 440
003911      000          KLBND=KLBND+1                                                      QAP7 441
003912      000    C                                                                        QAP7 442
003913      000          IF(KSNSW(21).EQ.1.AND.MOVE.EQ.0)WRITE(3,28)                        QAP7 443
003914      000          IF(KSNSW(21).EQ.1.AND.MOVE.EQ.1)WRITE(3,29)[illegible],LEVY        QAP7 444
003915      000          IF(KSNSW(21).EQ.1.AND.MOVE.EQ.2)WRITE(3,31)[illegible],LEVY        QAP7 445
003916      000    C                                                                        QAP7 446
003917      000    C                                                                        QAP7 447
003918      000          GO TO (130,222),KAM                                                QAP7 448
003919      000      130 NMIN1=NFCIL-1                                                      QAP7 449
003920      000          NMIN2=NLOCT-1                                                      QAP7 450
003921      000          LBTC=LBCST                                                         QAP7 451
003922      000          KEY=2                                                              QAP7 452
003923      000    C                                                                        QAP7 453
003924      000    C                                                                        QAP7 454
003925      000          DO 190 I=1,NFCIL                                                   QAP7 455
003926      000          DO 165 K=1,NFCIL                                                   QAP7 456
003927      000          WORK3(K)=F(I,K)                                                    QAP7 457
003928      000          WORK4(K)=K                                                         QAP7 458
003929      000      165 CONTINUE                                                           QAP7 459
003930      000          DO 180 J=1,NMIN1                                                   QAP7 460
003931      000          K=J+1                                                              QAP7 461
003932      000          DO 170 L=K,NFCIL                                                   QAP7 462
003933      000          IF(WORK3(J).GT.WORK3(L))GO TO 170                                  QAP7 463
003934      000          KEEP=WORK3(J)                                                      QAP7 464
003935      000          WORK3(J)=WORK3(L)                                                  QAP7 465
003936      000          WORK3(L)=KEEP                                                      QAP7 466
003937      000          KEEP=WORK4(J)                                                      QAP7 467
003938      000          WORK4(J)=WORK4(L)                                                  QAP7 468
003939      000          WORK4(L)=KEEP                                                      QAP7 469
003940      000      170 CONTINUE                                                           QAP7 470
003941      000      180 CONTINUE                                                           QAP7 471
003942      000          DO 185 J=1,NFCIL                                                   QAP7 472
003943      000      185 INTRI(I,J)=WORK4(J)                                                QAP7 473
003944      000          IF(KSNSW(23).EQ.1)WRITE(6,11)I,(WORK3(L),L=1,NFCIL)                QAP7 474
003945      000      190 CONTINUE                                                           QAP7 475
003946      000    C                                                                        QAP7 476
003947      000    C                                                                        QAP7 477
003948      000    C                                                                        QAP7 478
003949      000          DO 230 JJ=1,NLOCT                                                  QAP7 479
003950      000          DO 192 K=1,NLOCT                                                   QAP7 480
003951      000          WORK6(K)=D(JJ,K)                                                   QAP7 481
003952      000          IF(WORK6(K).EQ.0)WORK6(K)=9999                                     QAP7 482
003953      000          WORK4(K)=K                                                         QAP7 483
003954      000      192 CONTINUE                                                           QAP7 484
003955      000    C                                                                        QAP7 485
003956      000          DO 210 J=1,NMIN2                                                   QAP7 486
003957      000          K=J+1                                                              QAP7 487
003958      000          DO 200 L=K,NLOCT                                                   QAP7 488
```

```
003959      000          IF(WORK6(J).LT.WORK6(L))GO TO 200                                  QAP7 489
003960      000          KEEP=WORK6(J)                                                      QAP7 490
003961      000          WORK6(J)=WORK6(L)                                                  QAP7 491
003962      000          WORK6(L)=KEEP                                                      QAP7 492
003963      000          KEEP=WORK4(J)                                                      QAP7 493
003964      000          WORK4(J)=WORK4(L)                                                  QAP7 494
003965      000          WORK4(L)=KEEP                                                      QAP7 495
003966      000        200 CONTINUE                                                         QAP7 496
003967      000        210 CONTINUE                                                         QAP7 497
003968      000          IF(KSNSW(23).EQ.1)WRITE(6,12)JJ,(WORK6(L),L=1,NLOCT)             QAP7 498
003969      000          DO 220 K=1,NLOCT                                                   QAP7 499
003970      000          WORK2(JJ,K)=WORK6(K)                                               QAP7 500
003971      000        220 DIST1(JJ,K)=WORK4(K)                                             QAP7 501
003972      000        230 CONTINUE                                                         QAP7 502
003973      000    C                                                                        QAP7 503
003974      000    C                                                                        QAP7 504
003975      000    C   CONSTRUCT THE WORK1 MATRIX                                         QAP7 505
003976      000        222 DO 320 I=1,NFCIL                                                 QAP7 506
003977      000          IF(FCFLG(I))GO TO 300                                            QAP7 507
003978      000          KEEP=0                                                           QAP7 508
003979      000          DO 250 L=1,NFCIL                                                 QAP7 509
003980      000          LL=INTRI(I,L)                                                    QAP7 510
003981      000          IF(FCFLG(LL))GO TO 250                                           QAP7 511
003982      000          KEEP=KEEP+1                                                      QAP7 512
003983      000          WORK3(KEEP)=F(I,LL)                                              QAP7 513
003984      000        250 CONTINUE                                                       QAP7 514
003985      000          NFREE=KEEP                                                       QAP7 515
003986      000    C                                                                      QAP7 516
003987      000          DO 290 J=1,NLOCT                                                 QAP7 517
003988      000          IF(DANED(I,J))GO TO 260                                          QAP7 518
003989      000          IF(FLOGT(J))GO TO 260                                            QAP7 519
003990      000          KEEP=0                                                           QAP7 520
003991      000          DO 260 L=1,NLOCT                                                 QAP7 521
003992      000          LL=DIST1(J,L)                                                    QAP7 522
003993      000          IF(FLOGT(LL))GO TO 260                                           QAP7 523
003994      000          KEEP=KEEP+1                                                      QAP7 524
003995      000          WORK4(KEEP)=D(J,LL)                                              QAP7 525
003996      000        260 CONTINUE                                                       QAP7 526
003997      000          KEEP=0                                                           QAP7 527
003998      000          IF(NFREE.EQ.0)GO TO 272                                          QAP7 528
003999      000    C   HERE WE CALCULATE WHAT IS EQUIVALENT TO AN OLD IC3 ELEMENT        QAP7 529
004000      000          DO 270 L=1,NFREE                                                 QAP7 530
004001      000        270 KEEP=KEEP+(WORK3(L)*WORK4(L))                                  QAP7 531
004002      000    C   HERE WE CALCULATE WHAT IS EQUIVALENT TO AN OLD IC2 MATRIX ENTRY    QAP7 532
004003      000        272 LL=0                                                           QAP7 533
004004      000          DO 275 L=1,NFCIL                                                 QAP7 534
004005      000          IF(.NOT.FCFLG(L))GO TO 275                                       QAP7 535
004006      000          JL=LOCAT(L)                                                      QAP7 536
004007      000          LL=LL+(D(JL,J)*F(L,I))                                           QAP7 537
004008      000        275 CONTINUE                                                       QAP7 538
004009      000    C                                                                      QAP7 539
004010      000          WORK1(I,J)=KEEP+(2*LL)                                           QAP7 540
004011      000          GO TO 290                                                        QAP7 541
004012      000        280 WORK1(I,J)=32767                                               QAP7 542
004013      000        290 CONTINUE                                                       QAP7 543
004014      000          GO TO 320                                                        QAP7 544
004015      000    C                                                                      QAP7 545
```

```
004016  000      300 DO 310 L=1,NLOCT                                                    QAP7 546
004017  000      310 WORK1(I,L)=32767                                                    QAP7 547
004018  000          L=LOCAT(I)                                                          QAP7 548
004019  000          WORK1(I,L)=0                                                        QAP7 549
004020  000      320 CONTINUE                                                            QAP7 550
004021  000      C                                                                       QAP7 551
004022  000      C   CALCULATE THE REDUCTIONS                                            QAP7 552
004023  000          KEEP=0                                                              QAP7 553
004024  000          DO 350 I=1,NFCIL                                                    QAP7 554
004025  000          WORK3(I)=32767                                                      QAP7 555
004026  000          DO 330 J=1,NLOCT                                                    QAP7 556
004027  000          IF(WORK1(I,J).GE.WORK3(I))GO TO 330                                 QAP7 557
004028  000          WORK3(I)=WORK1(I,J)                                                 QAP7 558
004029  000      330 CONTINUE                                                            QAP7 559
004030  000          KEEP=KEEP+WORK3(I)                                                  QAP7 560
004031  000          DO 340 J=1,NLOCT                                                    QAP7 561
004032  000          WORK2(I,J)=WORK1(I,J)                                               QAP7 562
004033  000      340 WORK1(I,J)=WORK1(I,J)-WORK3(I)                                      QAP7 563
004034  000      350 CONTINUE                                                            QAP7 564
004035  000      C                                                                       QAP7 565
004036  000          DO 400 J=1,NLOCT                                                    QAP7 566
004037  000          WORK6(J)=32767                                                      QAP7 567
004038  000          DO 375 I=1,NFCIL                                                    QAP7 568
004039  000          IF(WORK1(I,J).GE.WORK6(J))GO TO 375                                 QAP7 569
004040  000          WORK6(J)=WORK1(I,J)                                                 QAP7 570
004041  000      375 CONTINUE                                                            QAP7 571
004042  000          DO 380 I=1,NFCIL                                                    QAP7 572
004043  000      380 WORK1(I,J)=WORK1(I,J)-WORK6(J)                                      QAP7 573
004044  000      400 CONTINUE                                                            QAP7 574
004045  000          IF(NFCIL.EQ.NLOCT)GO TO 410                                         QAP7 575
004046  000          DO 405 I=1,10                                                       QAP7 576
004047  000      405 WRITE(6,98)                                                         QAP7 577
004048  000       98 FORMAT(02X,2(' **** WARNING ****'),3X,' YOU NEED TO ARRANGE AN      QAP7 578
004049  000         * ARRAY IN LWBND2 SO INCLUDE THE SUBROUTINE')                        QAP7 579
004050  000          STOP                                                                QAP7 580
004051  000      410 DO 415 J=1,NFCIL                                                    QAP7 581
004052  000      415 KEEP=KEEP+WORK6(J)                                                  QAP7 582
004053  000          LBCST=KEEP                                                          QAP7 583
004054  000      C                                                                       QAP7 584
004055  000      C   PRINT THE MATRICES IF YOU WISH                                     QAP7 585
004056  000          IF(KSNSW(24).EQ.0)GO TO 450                                         QAP7 586
004057  000          WORK4(1)=9999                                                       QAP7 587
004058  000          DO 420 J=2,40                                                       QAP7 588
004059  000      420 WORK4(J)=J-1                                                        QAP7 589
004060  000          WRITE(6,16)                                                         QAP7 590
004061  000          WRITE(6,01)(WORK4(J),J=1,40)                                        QAP7 591
004062  000          WRITE(6,07)                                                         QAP7 592
004063  000          DO 430 I=1,NFCIL                                                    QAP7 593
004064  000      430 WRITE(6,01)I,(WORK2(I,J),J=1,NLOCT),WORK3(I)                        QAP7 594
004065  000          WRITE(6,07)                                                         QAP7 595
004066  000          WRITE(6,01)NFCIL,(WORK6(J),J=1,NLOCT),KEEP                          QAP7 596
004067  000          WRITE(6,07)                                                         QAP7 597
004068  000          WRITE(6,18)                                                         QAP7 598
004069  000          WRITE(6,01)(WORK4(J),J=1,40)                                        QAP7 599
004070  000          WRITE(6,07)                                                         QAP7 600
004071  000          DO 440 I=1,NFCIL                                                    QAP7 601
004072  000      440 WRITE(6,01)I,(WORK1(I,J),J=1,NLOCT),WORK3(I)                        QAP7 602
```

```
004073      000          WRITE(6,07)                                                GAP7 603
004074      000          WRITE(6,01)NFCIL,(WORK6(J),J=1,NLOCT),KEEP                 GAP7 604
004075      000          WRITE(6,07)                                                GAP7 605
004076      000    C                                                                GAP7 606
004077      000      450 IF(KSNSW(25).EQ.1)WRITE(6,25)KEEP                          GAP7 607
004078      000          IF(KEEP.GE.32767)GO TO 455                                 GAP7 608
004079      000          IF(KAM.EQ.1)GO TO 750                                      GAP7 609
004080      000          KOMP=0.5*FLOAT(KEEP)                                       GAP7 610
004081      000          KOMP=KOMP+IC1+IWD                                          GAP7 611
004082      000          IF(KOMP.LT.KTOT)GO TO 775                                  GAP7 612
004083      000          ITOT=0                                                     GAP7 613
004084      000          IF(KSNSW(28).EQ.1)WRITE(6,28)KOMP,KTOT                     GAP7 614
004085      000          NECTP=NECTP-1                                              GAP7 615
004086      000          GO TO 783                                                  GAP7 616
004087      000      455 LBCST=32767                                                GAP7 617
004088      000            IC2=32767                                                GAP7 618
004089      000           LTOT=3276700                                              GAP7 619
004090      000          GO TO 820                                                  GAP7 620
004091      000    C                                                                GAP7 621
004092      000    C   INITIALIZE HADY                                              GAP7 622
004093      000      750 N1=NFCIL                                                   GAP7 623
004094      000          IF(NFCIL.EQ.NLOCT)GO TO 760                                GAP7 624
004095      000    C   ADD ADUMMY FACILITY                                          GAP7 625
004096      000          N1=NFCIL+1                                                 GAP7 626
004097      000          DO 755 J=1,NLOCT                                           GAP7 627
004098      000      755 WORK1(N1,J)=0                                              GAP7 628
004099      000          IORIG(N1)=NLOCT-NFCIL                                      GAP7 629
004100      000      760 DO 765 I=1,NFCIL                                           GAP7 630
004101      000      765 IORIG(I)=1                                                 GAP7 631
004102      000          DO 770 J=1,NLOCT                                           GAP7 632
004103      000      770 IDEST(J)=1                                                 GAP7 633
004104      000          KEY=2                                                      GAP7 634
004105      000      773 CALL HADY(N1,K1,NROW(1,1),IDEST(1),IORIG(1),KEY)           GAP7 635
004106      000          GO TO(775,783),KEY                                         GAP7 636
004107      000      775 CALL GARY(N1,K1,NROW(1,1),NCOL(1),KEY)                     GAP7 637
004108      000          GO TO(773,780),KEY                                         GAP7 638
004109      000      780 WRITE(6,99)                                                GAP7 639
004110      000          RETURN                                                     GAP7 640
004111      000       99 FORMAT(02X,3(' ***** ERROR IN GARY ***'))                  GAP7 641
004112      000      783 NOCST=ITOT+LBCST                                           GAP7 642
004113      000          NECTP=NECTP+1                                              GAP7 643
004114      000          IF(KSNSW(28).EQ.1)WRITE(6,28)ITOT                          GAP7 644
004115      000    C                                                                GAP7 645
004116      000            IC2=0.5*FLOAT(NOCST)                                     GAP7 646
004117      000          KEY=1                                                      GAP7 647
004118      000          LTOT=0                                                     GAP7 648
004119      000    C   EVALUATE THE SOLUTION                                        GAP7 649
004120      000          DO 784 K=1,NLOCT                                           GAP7 650
004121      000      784 WORK8(K)=0                                                 GAP7 651
004122      000          DO 790 I=1,NFCIL                                           GAP7 652
004123      000          DO 785 K=1,NLOCT                                           GAP7 653
004124      000          IF(WORK8(K).EQ.1)GO TO 785                                 GAP7 654
004125      000          IF(JBAS(I,K).LE.K1)GO TO 785                               GAP7 655
004126      000          WORK4(I)=K                                                 GAP7 656
004127      000          WORK8(K)=1                                                 GAP7 657
004128      000          GO TO 790                                                  GAP7 658
004129      000      785 CONTINUE                                                   GAP7 659
```

```
      790 CONTINUE                                                          QAP7 660
C                                                                           QAP7 661
      DO 810 I=1,NFCIL                                                      QAP7 662
      DO 800 J=1,NFCIL                                                      QAP7 663
      LOCI=WORK4(I)                                                         QAP7 664
      LOCJ=WORK4(J)                                                         QAP7 665
  800 LTOT=LTOT+(F(I,J)*D(LOCI,LOCJ))                                       QAP7 666
  810 CONTINUE                                                              QAP7 667
      LTOT=0.5*FLOAT(LTOT)                                                  QAP7 668
      IF(KAM.EQ.1)KTOT=LTOT                                                 QAP7 669
C                                                                           QAP7 670
C   UPDATE IC1                                                              QAP7 671
  820 IC1=IC1+IWD                                                           QAP7 672
      IF(KAM.EQ.2)GO TO 830                                                 QAP7 673
      KEEP=IC1+IC2                                                          QAP7 674
      IF(KEEP.GT.LBTC )GO TO 830                                            QAP7 675
      DO 825 I=1,10                                                         QAP7 676
  825 WRITE(6,41)KEEP,LBTC                                                  QAP7 677
      LBCST=LBTC                                                            QAP7 678
      GO TO 840                                                             QAP7 679
  830 LBCST=IC1+IC2                                                         QAP7 680
C                                                                           QAP7 681
  840 IF(KSNSW(22).EQ.1)WRITE(6,30)IC1                                      QAP7 682
      IF(KSNSW(22).EQ.1)WRITE(6,30)IC2                                      QAP7 683
      IF(KSNSW(26).EQ.1)WRITE(6,32)LTOT,KTOT                                QAP7 684
      IF(KSNSW(29).EQ.1)WRITE(6,39)LBCST                                    QAP7 685
C                                                                           QAP7 686
      RETURN                                                                QAP7 687
C                                                                           QAP7 688
    1 FORMAT(10X,20I4)                                                      QAP7 689
    6 FORMAT(1H1)                                                           QAP7 690
    7 FORMAT(/)                                                             QAP7 691
   11 FORMAT(/10X,'INTERACTIONS FOR ELEMENT',I3,2X,20I4)                    QAP7 692
   12 FORMAT(/10X,'DISTANCES FOR LOCATION  ',I3,2X,20I4)                    QAP7 693
   14 FORMAT(1H+,110X,'PRODUCTS=',I7)                                       QAP7 694
   16 FORMAT(/02X,'MATRICES FOR LOWER BOUND FOLLOW,'/                       QAP7 695
     *       10X,'COMPLETE MATRIX',/)                                       QAP7 696
   18 FORMAT(10X,'REDUCED MATRIX',/)                                        QAP7 697
   19 FORMAT(100X,'ABSOLUTE LOWER BOUND=',I8)                               QAP7 698
   21 FORMAT(/10X,'MATRIX OF LOWER BOUNDS WHEN INCLUSIONS ARE ENFORCRD')    QAP7 699
   22 FORMAT(/10X,'MATRIX OF LOWER BOUNDS WHEN EXCLUSIONS ARE ENFORCRD')    QAP7 700
   23 FORMAT(/10X,'LOWER BOUND CALCULATED BY LP  =',I8)                     QAP7 701
   24 FORMAT(10X,'NOT NECESSARY TO SOLVE TP AS LB BY REDUCT=',I8,'  WHI      QAP7 702
     *E CURRENT UB=',I8)                                                    QAP7 703
   28 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT THE START')         QAP7 704
   29 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT ITER ',I3,'        QAP7 705
     *LEVEL',I3,' ON A FORWARD MOVE')                                       QAP7 706
   31 FORMAT(1H0,'   RESULTS FOR CALCULATIONS FOR LB AT ITER ',I3,'        QAP7 707
     *LEVEL',I3,' ON A BACKWARD MOVE')                                      QAP7 708
   32 FORMAT(90X,'VALUE IF C1 =',I5)                                        QAP7 709
   34 FORMAT(90X,'VALUE OF C2 =',I5)                                        QAP7 710
   36 FORMAT(90X,'VALUE OF UB =',I5,'   I7)                                 QAP7 711
   39 FORMAT(90X,'VALUE OF LB =',I5)                                        QAP7 712
   41 FORMAT(02X,4('**** NOOOOOOTYCE ****'),'  IC1+IC2=',I7,2X,'LBCST       QAP7 713
     *READ IN=',I7)                                                         QAP7 714
C                                                                           QAP7 715
C                                                                           QAP7 716
```

```
004187  000        END                                                          GAP7 717
004188  000  @FLT,SI TUMER-S*SINAN-GAP7,T:PUT,,,15257410377(1
004189  000        SUBROUTINE TIMOUT
004190  000        K=ITIME(I,J)
004191  000        T=I/5000.
004192  000        WRITE(6,2000)T
004193  000   2000 FORMAT(' TIME',F10.3,' SEC')
004194  000        RETURN
004195  000        END
004196  000
004197  000
004198  000  @ELT,SI TUMER-S*SINAN-GAIN,,,
```

```
004244      000          WRITE(6,8)                                                    GAIN  45
004245      000          DO 90 I=1,NFCIL                                               GAIN  46
004246      000       90 WRITE(6,01)(F(I,J),J=1,NFCIL)                                 GAIN  47
004247      000    C                                                                   GAIN  48
004248      000      110 NMIN1=NLOCT-1                                                 GAIN  49
004249      000          NMIN2=NFCIL-1                                                 GAIN  50
004250      000    C                                                                   GAIN  51
004251      000          DO 115 I=1,40                                                 GAIN  52
004252      000          LOCAT(I)=32767                                                GAIN  53
004253      000          DO 115 J=1,40                                                 GAIN  54
004254      000      115 MAKAN(I,J)=32767                                              GAIN  55
004255      000    C   CALCULATE THE DISTANCE FROM EACH LOCATION TO ALL ITHER LOVATIONS GAIN  56
004256      000          DO 130 I=1,NLOCT                                              GAIN  57
004257      000          WORK2(I)=0                                                    GAIN  58
004258      000          WORK3(I)=I                                                    GAIN  59
004259      000          DO 130 J=1,NLOCT                                              GAIN  60
004260      000      130 WORK2(I)=WORK2(I)+D(I,J)                                      GAIN  61
004261      000          IF(KSNSW(21).EQ.1)WRITE(6,11)(WORK2(I),I=1,NLOCT)             GAIN  62
004262      000          IF(KSNSW(21).EQ.1)WRITE(6,12)(WORK3(I),I=1,NLOCT)             GAIN  63
004263      000    C   RANK ALL THE LOCATIONS IN AN ASCENDING ORDER OF THE DISTANCES   GAIN  64
004264      000          DO 180 I=1,NMIN1                                              GAIN  65
004265      000          K=I+1                                                         GAIN  66
004266      000          DO 170 J=K,NLOCT                                              GAIN  67
004267      000          IF(WORK2(I).LT.WORK2(J))GO TO 170                             GAIN  68
004268      000          KEEP=WORK2(I)                                                 GAIN  69
004269      000          LEEP=WORK3(I)                                                 GAIN  70
004270      000          WORK2(I)=WORK2(J)                                             GAIN  71
004271      000          WORK3(I)=WORK3(J)                                             GAIN  72
004272      000          WORK2(J)=KEEP                                                 GAIN  73
004273      000          WORK3(J)=LEEP                                                 GAIN  74
004274      000      170 CONTINUE                                                      GAIN  75
004275      000      180 CONTINUE                                                      GAIN  76
004276      000          DO 190 J=1,NLOCT                                              GAIN  77
004277      000          K=WORK3(J)                                                    GAIN  78
004278      000          JFLAG(K)=J                                                    GAIN  79
004279      000      190 CONTINUE                                                      GAIN  80
004280      000    C                                                                   GAIN  81
004281      000    C                                                                   GAIN  82
004282      000          IF(KSNSW(22).EQ.0)GO TO 200                                   GAIN  83
004283      000          WRITE(6,21)                                                   GAIN  84
004284      000          WRITE(6,11)(WORK2(I),I=1,NLOCT)                               GAIN  85
004285      000          WRITE(6,12)(WORK3(I),I=1,NLOCT)                               GAIN  86
004286      000          WRITE(6,07)                                                   GAIN  87
004287      000          WRITE(6,14)(JFLAG(I),I=1,NLOCT)                               GAIN  88
004288      000    C                                                                   GAIN  89
004289      000    C   START RANKONG THE FACILITIES                                   GAIN  90
004290      000    C   A)CALCULATE THE NUMBER OF ELEMENTS AND CONNECTIONS WITH EACH ELEMENTGAIN  91
004291      000      200 DO 320 I=1,NFCIL                                            GAIN  92
004292      000          WORK7(I)=I                                                    GAIN  93
004293      000          WORK4(I)=I                                                    GAIN  94
004294      000          WORK5(I)=I                                                    GAIN  95
004295      000          E1(I)=0                                                       GAIN  96
004296      000          E2(I)=0                                                       GAIN  97
004297      000          DO 310 J=1,NFCIL                                              GAIN  98
004298      000          IF(I.EQ.J)GO TO 310                                           GAIN  99
004299      000          ISAV=F(I,J)                                                   GAIN 100
004300      000          IF(ISAV.LT.F(J,I))ISAV=F(J,I)                                 GAIN 101
```

```
004301  000          IF(ISAV.EQ.0)GO TO 310                                    GAIN 102
004302  000          E1(I)=E1(I)+1                                             GAIN 103
004303  000          E2(I)=E2(I)+ISAV                                          GAIN 104
004304  000        310 CONTINUE                                                GAIN 105
004305  000        320 CONTINUE                                                GAIN 106
004306  000      C                                                             GAIN 107
004307  000      C                                                             GAIN 108
004308  000          IF(KSNSW(23).EQ.0)GO TO 430                               GAIN 109
004309  000          WRITE(6,12)(WORK3(I),I=1,NFCIL)                           GAIN 110
004310  000          WRITE(6,17)(   E1(I),I=1,NFCIL)                           GAIN 111
004311  000          WRITE(6,18)(   E2(I),I=1,NFCIL)                           GAIN 112
004312  000      C                                                             GAIN 113
004313  000      C   RANK THE ELEMENTS IN A DESCENDING ORDER OF THE NUMBER OF ELEMENTS  GAIN 114
004314  000      C   CONNECTED TO EACH                                         GAIN 115
004315  000        330 DO 360 I=1,NMIN2                                        GAIN 116
004316  000          K=I+1                                                     GAIN 117
004317  000          DO 350 J=K,NFCIL                                          GAIN 118
004318  000          IF(E1(I).GT.E1(J))GO TO 350                               GAIN 119
004319  000          KEEP=E1(I)                                                GAIN 120
004320  000          LEEP=WORK4(I)                                             GAIN 121
004321  000          E1(I)=E1(J)                                               GAIN 122
004322  000          E1(J)=KEEP                                                GAIN 123
004323  000          WORK4(I)=WORK4(J)                                         GAIN 124
004324  000          WORK4(J)=LEEP                                             GAIN 125
004325  000        350 CONTINUE                                                GAIN 126
004326  000        360 CONTINUE                                                GAIN 127
004327  000      C                                                             GAIN 128
004328  000      C                                                             GAIN 129
004329  000          IF(KSNSW(24).EQ.0)GO TO 440                               GAIN 130
004330  000          WRITE(6,26)                                               GAIN 131
004331  000          WRITE(6,17)(   E1(I),I=1,NFCIL)                           GAIN 132
004332  000          WRITE(6,12)(WORK4(I),I=1,NFCIL)                           GAIN 133
004333  000      C                                                             GAIN 134
004334  000      C   RANK THE ELEMENTS IN A DESCENDING ORDER OF THE NUMBER OF CONNECTIONS  GAIN 135
004335  000        400 DO 460 I=1,NMIN2                                        GAIN 136
004336  000          K=I+1                                                     GAIN 137
004337  000          DO 450 J=K,NFCIL                                          GAIN 138
004338  000          IF(E2(I).GT.E2(J))GO TO 450                               GAIN 139
004339  000          KEEP=E2(I)                                                GAIN 140
004340  000          LEEP=WORK5(I)                                             GAIN 141
004341  000          E2(I)=E2(J)                                               GAIN 142
004342  000          E2(J)=KEEP                                                GAIN 143
004343  000          WORK5(I)=WORK5(J)                                         GAIN 144
004344  000          WORK5(J)=LEEP                                             GAIN 145
004345  000        450 CONTINUE                                                GAIN 146
004346  000        460 CONTINUE                                                GAIN 147
004347  000      C                                                             GAIN 148
004348  000          IF(KSNSW(25).EQ.0)GO TO 500                               GAIN 149
004349  000          WRITE(6,27)                                               GAIN 150
004350  000          WRITE(6,18)(   E2(I),I=1,NFCIL)                           GAIN 151
004351  000          WRITE(6,12)(WORK5(I),I=1,NFCIL)                           GAIN 152
004352  000      C   FLAG THE ELEMENTS                                         GAIN 153
004353  000        500 DO 530 I=1,NFCIL                                        GAIN 154
004354  000          K=WORK4(I)                                                GAIN 155
004355  000          R1(K)=I                                                   GAIN 156
004356  000          K=WORK5(I)                                                GAIN 157
004357  000          R2(K)=I                                                   GAIN 158
```

```
004356      000      500 CONTINUE                                                    GAIN 159
004359      000    C                                                                 GAIN 160
004360      000          IF(KSNSW(26).EQ.0)GO TO 550                                 GAIN 161
004361      000          WRITE(6,28)                                                 GAIN 162
004362      000          WRITE(6,12)(WORK7(I),I=1,NFCTL)                             GAIN 163
004363      000          WRITE(6,29)(   R1(I),I=1,NFCTL)                             GAIN 164
004364      000          WRITE(6,31)(   R2(I),I=1,NFCTL)                             GAIN 165
004365      000    C                                                                 GAIN 166
004366      000    C                                                                 GAIN 167
004367      000    C   E)FIND THE COMBINED RANK                                      GAIN 168
004368      000      580 DO 580 I=1,NFCIL                                            GAIN 169
004369      000          R(I)=R1(I)+R2(I)                                            GAIN 170
004370      000          WORK6(I)=I                                                  GAIN 171
004371      000      580 CONTINUE                                                    GAIN 172
004372      000          IF(KSNSW(26).EQ.1)WRITE(6,32)(R(I),I=1,NFCIL)               GAIN 173
004373      000    C                                                                 GAIN 174
004374      000    C   F)RANK THE ELEMENTS IN AN ASCENDING ORDER OF R                GAIN 175
004375      000          DO 600 I=1,NMIN2                                            GAIN 176
004376      000          K=I+1                                                       GAIN 177
004377      000          DO 590 J=K,NFCIL                                            GAIN 178
004378      000          IF(R(I).LT.R(J))GO TO 590                                   GAIN 179
004379      000          KEEP=R(I)                                                   GAIN 180
004380      000          LEEP=WORK6(I)                                               GAIN 181
004381      000          R(I)=R(J)                                                   GAIN 182
004382      000          WORK6(I)=WORK6(J)                                           GAIN 183
004383      000          R(J)=KEEP                                                   GAIN 184
004384      000          WORK6(J)=LEEP                                               GAIN 185
004385      000      590 CONTINUE                                                    GAIN 186
004386      000      600 CONTINUE                                                    GAIN 187
004387      000          DO 620 I=1,NFCIL                                            GAIN 188
004388      000          K=WORK6(I)                                                  GAIN 189
004389      000          IFLAG(K)=I                                                  GAIN 190
004390      000      620 CONTINUE                                                    GAIN 191
004391      000    C                                                                 GAIN 192
004392      000          IF(KSNSW(27).EQ.0)GO TO 650                                 GAIN 193
004393      000          WRITE(6,29)(   R1(I),I=1,NFCTL)                             GAIN 194
004394      000          WRITE(6,31)(   R2(I),I=1,NFCTL)                             GAIN 195
004395      000          WRITE(6,32)(    R(I),I=1,NFCTL)                             GAIN 196
004396      000          WRITE(6,07)                                                 GAIN 197
004397      000          WRITE(6,24)(IFLAG(I),I=1,NFCTL)                             GAIN 198
004398      000          WRITE(6,36)(WORK6(I),I=1,NFCTL)                             GAIN 199
004399      000    C                                                                 GAIN 200
004400      000    C                                                                 GAIN 201
004401      000    C   START ASSIGNING THE ELEMENTS                                 GAIN 202
004402      000      650 DO 660 I=1,NFCIL                                            GAIN 203
004403      000      660 FCFLG(I)=.FALSE.                                            GAIN 204
004404      000          DO 670 J=1,NLOCT                                            GAIN 205
004405      000      670 FLOGT(J)=.FALSE.                                            GAIN 206
004406      000    C                                                                 GAIN 207
004407      000    C                                                                 GAIN 208
004408      000    C   PICK  THE ELEMENT IN THE NEXT RANK , LOCATE IT , THEN PICK THE ONE  GAIN 209
004409      000    C   WITH MAXIMUM INTERACTIONS WITH IT , LOCATE IT SO AS TO MINIMIZE THE GAIN 210
004410      000    C   INCREASE IN TOTAL COST , CONTINUE UNTIL ALL ELEMENTS ARE LOCATED  GAIN 211
004411      000          DO 950 I=1,NFCIL                                            GAIN 212
004412      000          K=WORK6(I)                                                  GAIN 213
004413      000          IF(FCFLG(K))GO TO 950                                       GAIN 214
004414      000          DO 680 J=1,NLOCT                                            GAIN 215
```

```
                L=WORK3(J)                                                      GAIN 216
                IF(FLOGT(L))GO TO 680                                           GAIN 217
                LOCAT(K)=L                                                      GAIN 218
                FCFLG(K)=.TRUE.                                                 GAIN 219
                FLOGT(L)=.TRUE.                                                 GAIN 220
                IF(KSNSW(28).EQ.1)WRITE(6,[illegible])K,L                       GAIN 221
                IKNTR=1
                IF(I.EQ.1)GO TO 685
                KL=K
                GO TO 750                                                       GAIN 222
            680 CONTINUE                                                        GAIN 223
                WRITE(6,99)K                                                    GAIN 224
             99 FORMAT(02X,3('*** [illegible] ***'),'    NO PLACE FOR',I6)      GAIN 225
                GO TO 2000                                                      GAIN 226
C   PICK THE ELEMENT WITH MAXIMUM INTERACTIONS WITH K                           GAIN 227
            685 KEEP=-10                                                        GAIN 228
                LEEP=-10                                                        GAIN 229
                IKNTR=0
                DO 690 KK=1,NFCIL                                               GAIN 230
                IF(I.EQ.KK)GO TO 690                                            GAIN 231
                KL=WORK6(KK)                                                    GAIN 232
                IF(FCFLG(KL))GO TO 690                                          GAIN 233
                ISAV=F(K,KL)                                                    GAIN 234
                IF(F(KL,K).GT.ISAV)ISAV=F(KL,K)                                 GAIN 235
                IF(ISAV.LT.KEEP)GO TO 690                                       GAIN 236
                KEEP=ISAV                                                       GAIN 237
                LEEP=KL                                                         GAIN 238
            690 CONTINUE                                                        GAIN 239
                IF(LEEP.LT.0.AND.KSNSW(28).EQ.1)WRITE(6,[illegible])
             98 FORMAT(02X,3('*** [illegible] ***'),'  NO MORE ELEMENTS TO CHOSE  GAIN 241
               *FROM')                                                          GAIN 242
                IF(LEEP.LT.0)GO TO 750
                IF(LEEP.GT.0)GO TO 700                                          GAIN 243
                KL=K                                                            GAIN 244
                GO TO 750                                                       GAIN 245
C                                                                               GAIN 246
C   LOCATE THIS ELEMENT SO AS TO RESULT IN MINIMAL INCREASE IN COST             GAIN 247
            700 IWD=32767                                                       GAIN 248
                KL=LEEP                                                         GAIN 249
                DO 710 J=1,NLOCT                                                GAIN 250
                L=WORK3(J)                                                      GAIN 251
                IF(FLOGT(L))GO TO 710                                           GAIN 252
                KAM=0                                                           GAIN 253
                DO 705 II=1,NFCIL                                               GAIN 254
                LL=WORK6(II)                                                    GAIN 255
                IF(LL.EQ.KL)GO TO 705                                           GAIN 256
                IF(.NOT.FCFLG(LL))GO TO 705                                     GAIN 257
                LAM=LOCAT(LL)                                                   GAIN 258
                ISAV=F(KL,LL)                                                   GAIN 259
                IF(F(LL,KL).GT.F(KL,LL))ISAV=F(LL,KL)                           GAIN 260
                KAM=KAM+(ISAV*D(L,LAM))                                         GAIN 261
            705 CONTINUE                                                        GAIN 262
                IF(KAM.GT.IWD)GO TO 710                                         GAIN 263
                IWD=KAM                                                         GAIN 264
                LOCAT(KL)=L                                                     GAIN 265
            710 CONTINUE                                                        GAIN 266
                J=LOCAT(KL)                                                     GAIN 267
```

```
004474      000          FLOGT(J)=.TRUE.                                                    GAIN 268
004475      000          FCFLG(KL)=.TRUE.                                                   GAIN 269
004476      000          IF(KSNSW(28).EQ.1)WRITE(6,34)KL,J                                  GAIN 270
004477      000    C                                                                        GAIN 271
004478      000    C                                                                        GAIN 272
004479      000    C                                                                        GAIN 273
004480      000    C                                                                        GAIN 274
004481      000      750 KOMP=0                                                             GAIN 275
004482      000    C   EVALUATE THIS PARTIAL ASSIGNMENT                                   GAIN 276
004483      000          DO 770 II=1,NFCIL                                                  GAIN 277
004484      000          LL=WORK6(II)                                                       GAIN 278
004485      000          IF(.NOT.FCFLG(LL))GO TO 770                                        GAIN 279
004486      000          LAM=LOCAT(LL)                                                      GAIN 280
004487      000          DO 760 KK=1,NFCIL                                                  GAIN 281
004488      000          KM=WORK6(KK)                                                       GAIN 282
004489      000          IF(.NOT.FCFLG(KM))GO TO 760                                        GAIN 283
004490      000          KAM=LOCAT(KM)                                                      GAIN 284
004491      000          ISAV=F(LL,KM)                                                      GAIN 285
004492      000          IF(ISAV.LT.F(KM,LL))ISAV=F(KM,LL)                                  GAIN 286
004493      000          KOMP=KOMP+(ISAV*D(LAM,KAM))                                        GAIN 287
004494      000      760 CONTINUE                                                           GAIN 288
004495      000      770 CONTINUE                                                           GAIN 289
004496      000          KOMP=0.5*FLOAT(KOMP)                                               GAIN 290
004497      000    C                                                                        GAIN 291
004498      000          IF(KSNSW(29).EQ.1)WRITE(6,36)(LOCAT(II),II=1,35),KOMP              GAIN 292
004499      000    C                                                                        GAIN 293
004500      000    C   PERFORM ALL POSSIBLE INSERTIONS AND DELETIONS FOR LAST ELEMENT ONLY GAIN 294
004501      000          DO 775 II=1,NFCIL                                                  GAIN 295
004502      000      775 RESRV(II)=LOCAT(II)                                                GAIN 296
004503      000          KOUNT=0                                                            GAIN 297
004504      000          DO 790 II=1,NFCIL                                                  GAIN 298
004505      000          LL=WORK6(II)                                                       GAIN 299
004506      000          IF(LL.EQ.KL)GO TO 790                                              GAIN 300
004507      000          IF(.NOT.FCFLG(LL))GO TO 790                                        GAIN 301
004508      000          KOUNT=KOUNT+1                                                      GAIN 302
004509      000          KEY=0
004510      000      778 KEEJ=LOCAT(KL)
004511      000          KEEN=LOCAT(LL)
004512      000          IA=0
004513      000          DO 785 KR=1,NFCIL
004514      000          KK=WORK6(KR)
004515      000          IF(.NOT.FCFLG(KK))GO TO 785
004516      000          IF(KK.EQ.KL)GO TO 782
004517      000          IF(F(KK,KL).EQ.0)GO TO 782
004518      000          LAM=LOCAT(KK)
004519      000          IA=IA+(F(KK,KL)*D(KEEJ,LAM))
004520      000    C
004521      000      782 IF(KK.EQ.LL)GO TO 785
004522      000          IF(F(KK,LL).EQ.0)GO TO 785
004523      000          KAM=LOCAT(KK)
004524      000          IA=IA+(F(KK,LL)*D(KEEN,KAM))
004525      000      785 CONTINUE
004526      000          IF(KEY.EQ.1)GO TO 788
004527      000          IB=IA
004528      000          KEY=1
004529      000          KEEP=LOCAT(LL)                                                     GAIN 303
004530      000          LOCAT(LL)=LOCAT(KL)                                                GAIN 304
```

```
004529   000          LOCAT(KL)=KEEP                                          QAIN 305
004530   000          GO TO 778
004531   000       788 KEEN=    (IA-IB)
004532   000          KOST(KOUNT)=KOMP+KEEN
004533   000          DO 786 J=1,NFCIL                                        QAIN 324
004534   000       786 MAKAN(KOUNT,J)=LOCAT(J)                                QAIN 325
004535   000          LEEP=LOCAT(LL)                                          QAIN 326
004536   000          LOCAT(LL)=KEEP                                          QAIN 327
004537   000          LOCAT(KL)=LEEP                                          QAIN 328
004538   000    C                                                             QAIN 329
004539   000       790 CONTINUE                                               QAIN 330
004540   000    C                                                             QAIN 331
004541   000    C   WRITE THE RESULTS OF THE SINGLE INSERTIONS AND DELETIONS TEST IF YOU QAIN 332
004542   000    C   WISH                                                      QAIN 333
004543   000          IF(KSNSW(29).EQ.0)GO TO 850                             QAIN 334
004544   000          WRITE(6,37)                                             QAIN 335
004545   000          DO 810 J=1,KOUNT                                        QAIN 336
004546   000       810 WRITE(6,38)J,(MAKAN(J,I),I=1,NFCIL),KOST(J)            QAIN 337
004547   000        38 FORMAT(06X,I3,2X,[illegible],I6)                       QAIN 338
004548   000    C   CHECK FOR IMPROVEMENT                                     QAIN 339
004549   000       850 LEEP=KOMP                                              QAIN 340
004550   000          DO 850 J=1,KOUNT                                        QAIN 341
004551   000          IF(KOST(J) .GT.LEEP)GO TO 850                           QAIN 342
004552   000          LEEP=KOST(J)                                            QAIN 343
004553   000          KEEP=J                                                  QAIN 344
004554   000       850 CONTINUE                                               QAIN 345
004555   000    C                                                             QAIN 346
004556   000          IF(LEEP.EQ.KOMP)GO TO [illegible]                       QAIN 347
004557   000          IF(KSNSW(29).EQ.1) [illegible]KOMP,LEEP,KEEP)           QAIN 348
004558   000          KOMP=LEEP                                               QAIN 349
004559   000          DO 860 J=1,NFCIL                                        QAIN 350
004560   000       860 LOCAT(J)=MAKAN(KEEP,J)                                 QAIN 351
004561   000          IF(KSNSW(29).EQ.1)WRITE(6,[illegible])(LOCAT(J),J=1,NFCIL)
004562   000          GO TO 900                                               QAIN 353
004563   000    C                                                             QAIN 354
004564   000       870 IF(KSNSW(29).EQ.1)WRITE(6,[illegible])                 QAIN 355
004565   000          DO 880 J=1,NFCIL                                        QAIN 356
004566   000       880 LOCAT(J)=RESRV(J)                                      QAIN 357
004567   000    C                                                             QAIN 358
004568   000    C                                                             QAIN 359
004569   000       900 IF(IKHTR.EQ.1)GO TO 600                                QAIN 360
004570   000       950 CONTINUE                                               QAIN 361
004571   000    C                                                             QAIN 362
004572   000       970 WRITE(6,43)KOMP                                        QAIN 363
004573   000          WRITE(6,44)(LOCAT(I),I=1,NFCIL)                         QAIN 364
004574   000    C                                                             QAIN 365
004575   000    C                                                             QAIN 366
004576   000    C                                                             QAIN 367
004577   000    C   WRITE THE LOCATIONS OF THE ELEMENTS                       QAIN 368
004578   000          WRITE(6,09)                                             QAIN 369
004579   000          WRITE(6,16)(LOCAT(I),I=1,NFCIL)                         QAIN 370
004580   000    C                                                             QAIN 371
004581   000    C                                                             QAIN 372
004582   000    C   PERFORM THE TEST OF [illegible] AND DELETIONS IF DESIRED  QAIN 373
004583   000          IF(IROTA.EQ.0)GO TO 2000                                QAIN 374
004584   000          WRITE(6,51)                                             QAIN 375
004585   000          DO 1090 I=1,NFCIL                                       QAIN 376
```

```
004586  000    1090 KPLOC(I)=LOCAT(I)
004587  000         KMAX=0
004588  000         ITER=0
004589  000         KADD=0
004590  000         KURCS=KOMP
004591  000    1100 ITER=ITER+1
004592  000         DO 1300 IL=1,NFCIL
004593  000         DO 1105 II=1,NFCIL
004594  000    1105 RESRV(II)=LOCAT(II)
004595  000         ITHSN=0
004596  000         KL=WORK6(IL)
004597  000         KOUNT=0
004598  000         DO 1180 II=1,NFCIL
004599  000         LL=WORK6(II)
004600  000         IF(LL.EQ.KL)GO TO 1180
004601  000         KOUNT=KOUNT+1
004602  000         KEY=0
004603  000    1110 KEEJ=LOCAT(KL)
004604  000         KEEN=LOCAT(LL)
004605  000         IA=0
004606  000         DO 1130 KR=1,NFCIL
004607  000         KK=WORK6(KR)
004608  000         IF(KK.EQ.KL)GO TO 1120
004609  000         IF(F(KK,KL).EQ.0)GO TO 1120
004610  000         LAM=LOCAT(KK)
004611  000         IA=IA+(F(KK,KL)*D(KEEJ,LAM))
004612  000    C
004613  000    C
004614  000    1120 IF(KK.EQ.LL)GO TO 1130
004615  000         IF(F(KK,LL).EQ.0)GO TO 1130
004616  000         KAM=LOCAT(KK)
004617  000         IA=IA+(F(KK,LL)*D(KEEN,KAM))
004618  000    1130 CONTINUE
004619  000         IF(KEY.EQ.1)GO TO 1150
004620  000         IB=IA
004621  000         KEY=1
004622  000         KEEP=LOCAT(LL)
004623  000         LOCAT(LL)=LOCAT(KL)
004624  000         LOCAT(KL)=KEEP
004625  000         GO TO 1110
004626  000    1150 KEEN=(IA-IB)
004627  000         KOST(KOUNT)=KURCS+KEEN
004628  000         DO 1160  J=1,NFCIL
004629  000    1160 MAKAN(KOUNT,J)=LOCAT(J)
004630  000         LEEP=LOCAT(LL)
004631  000         LOCAT(LL)=KEEP
004632  000         LOCAT(KL)=LEEP
004633  000    C
004634  000    1180 CONTINUE
004635  000    C
004636  000    C   WRITE THE RESULTS OF THE TEST IF YOU WISH
004637  000         IF(KSNSW(31).EQ.0)GO TO 1220
004638  000         WRITE(6,37)
004639  000         DO 1200 J=1,KOUNT
004640  000    1200 WRITE(6,38)J,(MAKAN(J,KK),KK=1,35),KOST(J)
004641  000    C
004642  000    C   CHECK FOR IMPROVEMENTS
```

```
004643      000     1220 LEEP=32767
004644      000          DO 1240 J=1,KOUNT
004645      000          IF(KOST(J).GT.LEEP)GO TO 1240
004646      000          LEEP=KOST(J)
004647      000          KEEP=J
004648      000     1240 CONTINUE
004649      000          DO 1250 J=1,NFCIL
004650      000     1250 COMPR(IL,J)=MAKAN(KEEP,J)
004651      000          CSTCM(IL)=LEEP
004652      000     C
004653      000          IF(LEEP.GE.KOMP)GO TO 1270
004654      000          IF(KSNSW(32).EQ.1)WRITE(6,99)KOMP,LEEP,KEEP
004655      000          KOMP=LEEP
004656      000          DO 1260  J=1,NFCIL
004657      000          KPLOC( J)=MAKAN(KEEP,J)
004658      000     1260 LOCAT( J)=MAKAN(KEEP,J)
004659      000          IF(KSNSW(32).EQ.1)WRITE(6,98)(LOCAT(J),J=1,NFCIL)
004660      000          ITHSN=1
004661      000          KMAX=0
004662      000          GO TO 1500
004663      000     C
004664      000     1270 IF(KSNSW(32).EQ.1)WRITE(6,96)
004665      000          DO 1280 J=1,NFCIL
004666      000     1280 LOCAT(J)=RESRV(J)
004667      000     C
004668      000     1300 CONTINUE
004669      000     C
004670      000     1500 IF(ITHSN.EQ.1.AND.KSNSW(32).EQ.1)WRITE(6,95)
004671      000          KURCS=KOMP
004672      000           IF(ITHSN.EQ.1)GO TO 1700
004673      000     C
004674      000          IF(KSNSW(34).EQ.1)WRITE(6,94)
004675      000          IF(KSNSW(34).EQ.1)WRITE(6,98)(LOCAT(I),I=1,NFCIL)
004676      000          IF(KSNSW(34).EQ.1)WRITE(6,93)KEEP,KOMP
004677      000          KEEP=-10
004678      000          LEEP=32767
004679      000          DO 1530 I=1,NFCIL
004680      000          IF(CSTCM(I).GE.LEEP)GO TO 1530
004681      000          IF(CSTCM(I).EQ.KOMP)GO TO 1520
004682      000          IF(KADD.EQ.0)GO TO 1530
004683      000          DO 1510 II=1,KADD
004684      000          KEY=0
004685      000          DO 1505 J=1,NFCIL
004686      000          IF(ARRAY(II,J).EQ.COMPR(I,J))KEY=KEY+1
004687      000     1505 CONTINUE
004688      000          IF(KEY.NE.NFCIL)GO TO 1510
004689      000          GO TO 1530
004690      000     1510 CONTINUE
004691      000     1520 LEEP=CSTCM(I)
004692      000          KEEP=I
004693      000     1530 CONTINUE
004694      000          IF(KEEP.GT.0)GO TO 1535
004695      000          WRITE(6,97)
004696      000      97  FORMAT(10X,'HELP ME ,IAM STUCK')
004697      000          GO TO 1800
004698      000     1535 KMAX=KMAX+1
004699      000          IF(KMAX.GT.MAXM)GO TO 1800
```

```
004700  000          KADD=KADD+1
004701  000          IF(KADD.LE.90)GO TO 1537
004702  000          WRITE(6,96)
004703  000       96 FORMAT(10X,'RAN OUT OF ARRAY SPACE')
004704  000          GO TO 1800
004705  000     1537 DO 1540 I=1,NFCIL
004706  000          ARRAY(KADD,I)=COMPR(KEEP,I)
004707  000     1540 LOCAT(I)=COMPR(KEEP,I)
004708  000          KURCS=CSTCM(KEEP)
004709  000          IF(KSNSW(33).EQ.1)WRITE(6,54)(LOCAT(I),I=1,NFCIL)
004710  000          IF(KSNSW(33).EQ.1)WRITE(6,55)KURCS
004711  000          GO TO 1100
004712  000     C
004713  000     C
004714  000     1800 WRITE(6,9)
004715  000          WRITE(6,16)(KPLOC(I),I=1,NFCTL)
004716  000          WRITE(6,53)ITER,KOMP
004717  000     C
004718  000     2000 IF(MORE)3000,3000,10                                          QAIN 368
004719  000     3000 STOP                                                          QAIN 369
004720  000     C                                                                  QAIN 401
004721  000     C                                                                  QAIN 370
004722  000        1 FORMAT(10X,40I3)                                              QAIN 371
004723  000        2 FORMAT(10X,3('*** INITIAL  **))                               QAIN 372
004724  000        4 FORMAT(1H1,'   DATA FOR PROBLEM',I5/                          QAIN 373
004725  000         *       10X,'      NUMBER OF FACILITIES   =',I4/               QAIN 374
004726  000         *       10X,'      NUMBER OF LOCATIONS    =',I4//)             QAIN 375
004727  000        5 FORMAT(10X,'    DISTANCE MATRIX'/)                           QAIN 376
004728  000        6 FORMAT(1H1)                                                   QAIN 377
004729  000        7 FORMAT(/)                                                     QAIN 378
004730  000        8 FORMAT(/10X,'INTERACTION MATRIX'/)                            QAIN 379
004731  000        9 FORMAT(/10X,'LOCATIONS OF THE  FACILITIES ARE')               QAIN 380
004732  000       11 FORMAT('  DISTANCES    ',20I3)                               QAIN 381
004733  000       12 FORMAT('  INDICES      ',20I3)                               QAIN 382
004734  000       14 FORMAT(/2X,'POSITIONS OF  LOCATIONS IN THE RANK',30I3/36X,10I3)  QAIN 383
004735  000       16 FORMAT('  LOCATIONS    ',20I3)                               QAIN 384
004736  000       17 FORMAT('  NO OF ELEMENTS',20I3)                             QAIN 385
004737  000       18 FORMAT('  NO OF CONSTRAINTS',20I3)                          QAIN 386
004738  000       21 FORMAT(1H0,'ARRANGED LISTS OF  AND THEIR INDICES FOLLOW'/)   QAIN 387
004739  000       24 FORMAT(/2X,'POSITIONS OF  ELEMENTS IN THE RANK',30I3/36X,10I3)  QAIN 388
004740  000       26 FORMAT(/10X,'ARRAY OF RANK  ELEMENTS FOLLOWS'/)              QAIN 389
004741  000       27 FORMAT(/10X,'ARRAY OF RANK  CONSTRAINTS FOLLOWS'/)           QAIN 390
004742  000       28 FORMAT(/10X,'DISPLAY OF RELATIVE RANKS'/)                    QAIN 391
004743  000       29 FORMAT('          R1    ',20I3)                              QAIN 392
004744  000       31 FORMAT('          R2    ',20I3)                              QAIN 393
004745  000       32 FORMAT('          R     ',20I3)                              QAIN 394
004746  000       33 FORMAT(10X,'**MAJOR ***  , ELEMENT',I3,'  FIXED IN',I3)     QAIN 395
004747  000       34 FORMAT(10X,'**MINOR ***  , ELEMENT',I3,'  FIXED IN',I3)     QAIN 396
004748  000     C                                                                  QAIN 398
004749  000       36 FORMAT('  CONT OF POSIT',36I3)                              QAIN 397
004750  000       37 FORMAT(15X,'POSSIBLE ASSIGNMENTS')                           QAIN 400
004751  000       39 FORMAT(10X,'POSSIBLE IMPROVEMENT , ORIG COST=',I6,'  NEW COST=',  QAIN 402
004752  000         *I6,'     ASSOCIATED WITH LOCATION',I3,'  ABOVE')             QAIN 403
004753  000       41 FORMAT(20X,'CORRESPONDING PARTIAL ASSIGNMENT'/20X,35I3)      QAIN 404
004754  000     C                                                                  QAIN 399
004755  000       42 FORMAT(10X,'NO IMPROVEMENT POSSIBLE')                        QAIN 405
004756  000       43 FORMAT(/10X,'BEST INITIAL SOLUTION S COST=',I6)              QAIN 406
```

```
004757  000      44 FORMAT(/15X,'CORRESPONDING LOCATIONS ARE',5X,2013)                    GAIN 407
004758  000      46 FORMAT(/05X,'ORIGINAL FACTORIAL ASSIGNMENT'/(5X,2013,04X,'ITS COST=',GAIN 408
004759  000        *,I6)                                                                  GAIN 409
004760  000  C
004761  000      51 FORMAT(/10X,'RESULT OF CONVERGENT ASSIGNMENT')
004762  000      52 FORMAT(/10X,3('*'),' THIS IS AN IMPROVEMENT,
004763  000        1WE RESTART AGAIN.')
004764  000      53 FORMAT(/10X,'NUMBER OF ITERATIONS =',I8,'   COST=',I8)
004765  000      54 FORMAT(02X,'WE START FROM',2013)
004766  000      56 FORMAT(/9X,'WHOSE COST=',I8)
004767  000         END                                                                   GAIN 411
004768  000  @ELT,SI TUMER-S*SINAN-GAJULSTATD,,,13/27061112
004769  000  C    A SUBROUTINE TO READ [illegible] QUADRATIC ASSIGNMENT PROBLEM         GAIN 412
004770  000  C    THIS VERSION IS INTELLIGENT ET AU DATA                                GAIN 413
004771  000         SUBROUTINE DTAIN(JOB,...,TROTA)                                     GA
004772  000         COMMON IC1,IC2,IC3,...,XM,LCOST                                   GAIN 415
004773  000         COMMON MSTER(03400),...,ISN(2,40)                                 GAIN 416
004774  000         INTEGER   F(40,40),...(40),MSTER(...                              GAIN 417
004775  000         INTEGER   WORK1(40,80)                                            GAIN 418
004776  000         EQUIVALENCE( MSTER(...),...(1,1))                                 GAIN 419
004777  000         EQUIVALENCE( MSTER(...),...(1,1))                                 GAIN 420
004778  000         EQUIVALENCE( MSTER(...),...(1))                                   GAIN 421
004779  000         EQUIVALENCE( MSTER(...),...(1))                                   GAIN 422
004780  000         EQUIVALENCE( IWORK(...),WORK1(1,1))                               GAIN 423
004781  000  C                                                                        GAIN 424
004782  000  C    READ THE NUMBER OF FACILITIES , THE NUMBER OF LOCATIONS , THE FRACTIONGAIN 425
004783  000  C    OF THE INTERVAL (ITOT...) , THE NUMBER OF FIXED FACILITIES , AND  GAIN 426
004784  000  C    THE CONTINUATION PARAMETER                                        GAIN 427
004785  000         READ(5,1)JOB,NFCIL,...,TROTA,NFIXD,...,ITERM                     AIN 428
004786  000  C                                                                        GAIN 429
004787  000         DO 10 I=1,40                                                      GAIN 430
004788  000         DO 10 J=1,40                                                      GAIN 431
004789  000         F(I,J)=0                                                          GAIN 432
004790  000      10 D(I,J)=0                                                          GAIN 433
004791  000  C                                                                        GAIN 434
004792  000         DO 20 I=1,NFCIL                                                   GAIN 435
004793  000      20 READ(5,2)(WORK1(I,J),J=1,NFCIL)                                   GAIN 436
004794  000  C                                                                        GAIN 437
004795  000         DO 30 I=1,NFCIL                                                   GAIN 438
004796  000         DO 25 J=1,NFCIL                                                   GAIN 439
004797  000         IF(I.EQ.J)GO TO 25                                                GAIN 440
004798  000         IF(I.GT.J)F(I,J)=WORK1(I,J)                                       GAIN 441
004799  000         IF(I.LT.J)D(I,J)=WORK1(I,J)                                       GAIN 442
004800  000      25 CONTINUE                                                          GAIN 443
004801  000      30 CONTINUE                                                          GAIN 444
004802  000  C                                                                        GAIN 445
004803  000         DO 40 I=1,NFCIL                                                   GAIN 446
004804  000         DO 40 J=1,NFCIL                                                   GAIN 447
004805  000         F(I,J)=F(J,I)                                                     GAIN 448
004806  000         D(J,I)=D(I,J)                                                     GAIN 449
004807  000      40 CONTINUE                                                          GAIN 450
004808  000  C                                                                        GAIN 451
004809  000         DO 50 I=1,NFCIL                                                   GAIN 452
004810  000         DO 45 J=1,NFCIL                                                   GAIN 453
004811  000      45 CONTINUE                                                          GAIN 455
004812  000      50 CONTINUE                                                          GAIN 456
004813  000         IF(NFIXD.EQ.0)GO TO 80
```

```
004814      000   C                                                                      QAIN 457
004815      000         DO 60 I=1,NFIXD                                                  QAIN 458
004816      000      60 READ(5,1)  FIX(I),WHERE(I)                                       QAIN 459
004817      000   C                                                                      QAIN 460
004818      000      80 DO 999 I=1,60                                                    QAIN 461
004819      000     999 KSNSW(I)=0                                                       QAIN 462
004820      000         DO 998 I=28,40
004821      000     998 KSNSW(I)=0
004822      000   C                                                                      QAIN 463
004823      000         RETURN                                                           QAIN 464
004824      000   C                                                                      QAIN 465
004825      000   C                                                                      QAIN 466
004826      000   C                                                                      QAIN 467
004827      000       1 FORMAT(20X,12I5)                                                 QAIN 468
004828      000       2 FORMAT(10X,35I2)                                                 QAIN 469
004829      000         END                                                              QAIN 470
004830      000   @ELT,SI TUMER-S*SINAN-QAIN.TIMOUT,,,152574103711
004831      000         SUBROUTINE TIMOUT
004832      000         K=ITIME(I,J)
004833      000         T=I/5000.
004834      000         WRITE(6,2000)T
004835      000    2000 FORMAT(' TIME',F10.3,' SEC')
004836      000         RETURN
004837      000         END
004838      000   @ELT,SI TUMER-S*NEWQAP.MAIN,,,152574060412
004839      000   243
004840      000   C    A NEW PROGRAM FOR THE QUADRATIC ASSIGNMENT PROBLEM
004841      000         COMMON NFCIL,NLOCT,LFACT,LOCAT,
004842      000         COMMON MSTER(05200),IWORK(0060),KSNSW(0060)
004843      000         COMMON NKNAP,KHADY,KOUNTY,NUMV
004844      000   C
004845      000         INTEGER F(20,20),D(20,20),U(20,20),V(20,20),JDAS(20,20),ALPHA(20)
004846      000         LOGICAL Y(20,20),YBEST(20,20)
004847      000         INTEGER WORK1(20,20),WORK2(20,20)
004848      000         INTEGER UBEST(20,20),VBEST(20,20)
004849      000         INTEGER INDEX(0400),KOUNT(0400),JNDEX(0400),LNDEX(0400)
004850      000         INTEGER CNVXU(20,20),CNVXV(20,20)
004851      000   C
004852      000         EQUIVALENCE( MSTER(0001),    F(1,1))
004853      000         EQUIVALENCE( MSTER(0401),    D(1,1))
004854      000         EQUIVALENCE( MSTER(0801),    U(1,1))
004855      000         EQUIVALENCE( MSTER(1201),    V(1,1))
004856      000         EQUIVALENCE( MSTER(1601),YBEST(1,1))
004857      000         EQUIVALENCE( MSTER(2001),VBEST(1,1))
004858      000         EQUIVALENCE( MSTER(2401),UBEST(1,1))
004859      000         EQUIVALENCE( MSTER(2801),CNVXV(1,1))
004860      000         EQUIVALENCE( MSTER(3201),CNVXU(1,1))
004861      000         EQUIVALENCE( MSTER(3601),WORK2(1,1))
004862      000         EQUIVALENCE( MSTER(4001),WORK1(1,1))
004863      000         EQUIVALENCE( MSTER(4401),    Y(1,1))
004864      000         EQUIVALENCE( MSTER(4801),BEST(1,1))
004865      000   C
004866      000         EQUIVALENCE( IWORK(0001),WORK1(1,1))
004867      000         EQUIVALENCE( IWORK(0401),WORK2(1,1))
004868      000         EQUIVALENCE( IWORK(1001),JDAS(1,1))
004869      000         EQUIVALENCE( IWORK(1401),CNVXU(1,1))
004870      000   C
```

```
004871      000   C
004872      000   C   INITIALIZE VARIOUS COUNTERS AND ARRAYS
004873      000      10  ITER=+1
004874      000          IUBCS=+3276700
004875      000          LBCST=-3276700
004876      000          KOUNT=0
004877      000   C
004878      000          LIMIT=10
004879      000          NKNAP=0
004880      000          KHADY=0
004881      000          KGARY=0
004882      000           KUVW=0
004883      000   C
004884      000          DO 190 I=1,0400
004885      000     190 INDEX(I)=99
004886      000   C
004887      000   C
004888      000   C   READ THE KEYS CONTROLLING [illegible] PRINTING IN THE MAIN PROGRAM
004889      000          READ(5,02)(KSNSW(I),[illegible]
004890      000   C   READ THE KEYS CONTROLLING [illegible] PRINTING OF UVW
004891      000          READ(5,02)(KSNSW(I),[illegible])
004892      000   C
004893      000   C   READ THE PROBLEM DATA
004894      000          CALL DTAIN(JOB,LAMDA,[illegible],TTERM,IAVEC,[illegible])
004895      000   C
004896      000   C   PRINT THE DATA IF YOU WISH
004897      000          IF(KSNSW(01).EQ.0)GO TO 270
004898      000          WRITE(6,03)  JOB,[illegible]
004899      000          WRITE(6,04)LAMDA,[illegible]
004900      000     270 IF(KSNSW(02).EQ.0)GO TO [illegible]
004901      000          WRITE(6,08)
004902      000          DO 275 I=1,NFCIL
004903      000     275 WRITE(6,02)(F(I,J),J=1,[illegible])
004904      000          WRITE(6,09)
004905      000          DO 280 I=1,NLOCT
004906      000     280 WRITE(6,02)(D(I,J),J=1,[illegible])
004907      000          WRITE(6,07)
004908      000   C   GET THE STARTING VALUES OF U'S,V'S&W'S
004909      000     290 CALL UVW(1)
004910      000          DO 293 I=1,NFCIL
004911      000          DO 293 J=1,NLOCT
004912      000          CNVXV(I,J)=0
004913      000     293 CNVXU(I,J)=0
004914      000          AMU=.1*FLOAT(MEW)
004915      000   C
004916      000   C
004917      000   C   CALCULATE THE VALUES OF ALPHA
004918      000          DO 298 I=1,NFCIL
004919      000          KEEP=0
004920      000          DO 295 K=1,NFCIL
004921      000          IF(F(I,K).EQ.0)GO TO 295
004922      000          KEEP=KEEP+1
004923      000     295 CONTINUE
004924      000          ALPHA(I)=KEEP
004925      000     298 CONTINUE
004926      000   C
004927      000   C   SOLVE THE NFCIL*(NFCIL-1) KNAPSACK PROBLEMS
```

```
004928      000      300 KAM=0
004929      000          KOUNT=KOUNT+1
004930      000          KEEP=0
004931      000          DO 320 I=1,NFCIL
004932      000          DO 310 K=1,NFCIL
004933      000          IF(I.EQ.K)GO TO 310
004934      000          IF(F(I,K).EQ.0)GO TO 310
004935      000    C
004936      000          CALL KNAP(I,K,JEEP,LEEP,KOST)
004937      000          IF(KAM.LT.0400)GO TO 305
004938      000          WRITE(6,98)
004939      000          GO TO 1500
004940      000      305 KAM=KAM+1
004941      000          INDEX(KAM)=I
004942      000          KNDEX(KAM)=K
004943      000          JNDEX(KAM)=JEEP
004944      000          LNDEX(KAM)=LEEP
004945      000          KEEP=KEEP+KOST
004946      000          IF(KSNSW(06).EQ.0)GO TO 310
004947      000          WRITE(6,16)ITER,I,K,I,JEEP,K,LEEP,KOST
004948      000      310 CONTINUE
004949      000      320 CONTINUE
004950      000          IF(KSNSW(06).EQ.1)WRITE(6,17)ITER,KEEP
004951      000    C
004952      000    C   SOLVE THE ASSIGNMENT PROBLEM
004953      000    C   A)CONSTRUCT THE COST MATRIX
004954      000          IF(KSNSW(07).EQ.1)WRITE(6,18)ITER
004955      000          DO 350 I=1,NFCIL
004956      000          DO 340 J=1,NLOCT
004957      000      340 WORK1(I,J)=-ALPHA(I)*(U(I,J)+V(I,J))
004958      000          IF(KSNSW(07).EQ.1)WRITE(6,19)(WORK1(I,J),J=1,NLOCT)
004959      000      350 CONTINUE
004960      000    C
004961      000    C   B)CALL THE ASSIGNMENT ROUTINE
004962      000          KEY=1
004963      000          K1=30
004964      000      360 CALL HADY(NFCIL,K1,NROW1,NCOL1,ICOST,KEY)
004965      000          GO TO(380,410),KEY
004966      000      380 CALL GARY(NFCIL,K1,NROW1,NCOL1,KEY)
004967      000          GO TO(360,390),KEY
004968      000      390 WRITE(6,99)
004969      000          GO TO 1500
004970      000    C
004971      000      410 THETA=KEEP+ICOST
004972      000          THETA=THETA/2000000.
004973      000          IF(KSNSW(11).EQ.1)WRITE(6,11)ITER,KEEP,ICOST,THETA
004974      000          IF(THETA.LT.LBCST)GO TO 430
004975      000          LBCST=THETA
004976      000          KOUNT=0
004977      000    C
004978      000    C   STORE THE BEST VALUES OF THE U'S AND V'S
004979      000          DO 420 I=1,NFCIL
004980      000          DO 420 J=1,NLOCT
004981      000          UBEST(I,J)=U(I,J)
004982      000      420 VBEST(I,J)=V(I,J)
004983      000    C
004984      000    C
```

```
004985      000   C **EVALUATE THE PRIMAL SOLUTION OUT OF THE ASSIGNMENT ALGORITHM
004986      000   C   A)INITIALIZE THE Y'S
004987      000     450 DO 455 I=1,NFCIL
004988      000         DO 455 J=1,NLOCT
004989      000         Y(I,J)=.FALSE.
004990      000     455 CONTINUE
004991      000   C
004992      000   C
004993      000   C   B)CALCULATE THE QUADRATIC COST
004994      000         KEEP=IUBCS
004995      000         CALL UVW(2)
004996      000         IF(KSNSW(12).EQ.1)WRITE(6,12)IT2,IUBCS,KEEP
004997      000   C
004998      000         IF(IUBCS.GE.KEEP)GO TO 480
004999      000   C   C)CALCULATE THE BEST SOLUTION VECTOR
005000      000         DO 470 I=1,NFCIL
005001      000         DO 465 J=1,NLOCT
005002      000         YBEST(I,J)=.FALSE.
005003      000         IF(Y(I,J))YBEST(I,J)=.TRUE.
005004      000     465 CONTINUE
005005      000     470 CONTINUE
005006      000         GO TO 490
005007      000     480 IUBCS=KEEP
005008      000     490 IF(LBCST.GE.(IUBCS-I[illegible])GO TO 900
005009      000   C
005010      000   C
005011      000         KEY=0
005012      000   C **UPDATE THE VALUES OF U'S,V'S AND W'S
005013      000   C   A)CALCULATE THE U'S
005014      000         IT=0
005015      000         IF(KSNSW(16).EQ.1)WRITE([illegible])
005016      000         DO 540 I=1,NFCIL
005017      000         DO 530 J=1,NLOCT
005018      000         KEEP=0
005019      000         DO 515 K=1,NFCIL
005020      000         IF(I.EQ.K)GO TO 515
005021      000         IF(F(I,K).EQ.0)GO TO 515
005022      000         DO 514 L=1,NLOCT
005023      000         IF(J.EQ.L)GO TO 514
005024      000         KEEP=KEEP+X(I,J,K,L)
005025      000     514 CONTINUE
005026      000     515 CONTINUE
005027      000         KAM=0
005028      000         IF(Y(I,J))KAM=ALPHA(I)
005029      000         KEEP=KEEP-KAM
005030      000         IF(KEEP.NE.0)KEY=1
005031      000         IF(KSNSW(16).EQ.1)WRITE(6,29)I,J,KEEP
005032      000         KEEPA=IABS(KEEP)
005033      000         IT=IT+KEEPA
005034      000         WORK1(I,J)=KEEP
005035      000     530 CONTINUE
005036      000     540 CONTINUE
005037      000   C
005038      000   C   B)UPDATE THE V'S
005039      000         DO 560 K=1,NFCIL
005040      000         DO 570 L=1,NLOCT
005041      000         KEEP=0
```

```
005042      000          DO 555 I=1,NFCIL
005043      000          IF(K.EQ.I)GO TO 555
005044      000          IF(F(K,I).EQ.0)GO TO 555
005045      000          DO 554 J=1,NLOCT
005046      000          IF(J.EQ.L)GO TO 554
005047      000          KEEP=KEEP+X(I,J,K,L)
005048      000        554 CONTINUE
005049      000        555 CONTINUE
005050      000          KAM=0
005051      000          IF(Y(K,L))KAM=ALPHA(K)
005052      000          KEEP=KEEP-KAM
005053      000          IF(KSNSW(16).EQ.1)WRITE(6,29)K,L,KEEP
005054      000          KEEPA=IABS(KEEP)
005055      000          IT=IT+KEEPA
005056      000          WORK2(K,L)=KEEP
005057      000          IF(KEEP.NE.0)KEY=1
005058      000        570 CONTINUE
005059      000        580 CONTINUE
005060      000          IF(IT.EQ.0)GO TO 800
005061      000          IT=1*IT
005062      000          DO 600 I=1,NFCIL
005063      000          DO 600 J=1,NLOCT
005064      000          ADD=1000000.*FLOAT(WORK1(I,J))*LAMDA/FLOAT(IT)
005065      000          IF(IAVER.EQ.0)GO TO 595
005066      000          T=AMU*FLOAT(WORK1(I,J))*((1.-ANU)*CNVXU(I,J))
005067      000          ADD=1000000.*T*LAMDA/FLOAT(IT)
005068      000          CNVXU(I,J)=T
005069      000      595 KDD=ADD+.5
005070      000          U(I,J)=U(I,J)+KDD
005071      000        600 CONTINUE
005072      000          DO 610 K=1,NFCIL
005073      000          DO 610 L=1,NLOCT
005074      000          ADD=1000000.*FLOAT(WORK2(K,L))*LAMDA/FLOAT(IT)
005075      000          IF(IAVER.EQ.0)GO TO 605
005076      000          T=AMU*FLOAT(WORK2(K,L))*((1.-ANU)*CNVXV(K,L))
005077      000          ADD=1000000.*T*LAMDA/FLOAT(IT)
005078      000          CNVXV(K,L)=T
005079      000        605 KDD=ADD+.5
005080      000          V(K,L)=V(K,L)+KDD
005081      000        610 CONTINUE
005082      000      C
005083      000      C   C)PRINT THE U'S AND V'S IF YOU WISH
005084      000          IF(KSNSW(14).EQ.0)GO TO 640
005085      000          WRITE(6,14)
005086      000          DO 620 I=1,NFCIL
005087      000        620 WRITE(6,01)(U(I,J),J=1,NLOCT)
005088      000          WRITE(6,07)
005089      000          DO 630 I=1,NFCIL
005090      000        630 WRITE(6,01)(V(I,J),J=1,NLOCT)
005091      000      C
005092      000      C **TEST WHETHER PRIMAL FEASIBILITY IS ACHIEVED
005093      000        640 IF(KEY.EQ.0)GO TO 800
005094      000          IF(KOUNT.LT.ITERM)GO TO 660
005095      000          WRITE(6,23)ITER,LBCST,IUBCS
005096      000          GO TO 1000
005097      000      C
005098      000      C
```

```
      600 ITER=ITER+1
          IF(ITER.LT.LIMIT)GO TO 605
          LIMIT=LIMIT+35
          ALAMD=LAMDA
          ALAMD=.5*ALAMD
          LAMDA=ALAMD+.5
          WRITE(6,31)LAMDA
       31 FORMAT(10X,'LAMDA=',F8.5)
          IF(LAMDA.LT.1)LAMDA=1
      605 IF(ITER.GT.200)GO TO 670
          GO TO 300
C
      670 WRITE(6,27)ITER,LBCST,IUBCS
          GO TO 1000
C
C
      800 WRITE(6,21)ITER
          WRITE(6,22)IUBCS
          GO TO 1000
C
      900 WRITE(6,24)ITER,LBCST,IUBCS,ITOP
C
C   PRINT THE VALUES OF THE U AND V VARS
 1000 WRITE(6,14)
          DO 1010 I=1,NFCIL
 1010 WRITE(6,01)(UBEST(I,J),J=1,NLOCT)
          WRITE(6,07)
C
          DO 1020 I=1,NFCIL
 1020 WRITE(6,01)(VBEST(I,J),J=1,NLOCT)
          WRITE(6,07)
C
C
C
C   EDIT THE BEST SOLUTION TO THE ASSIGNMENT PROBLEM
 1100 WRITE(6,26)
          DO 1120 I=1,NFCIL
          DO 1110 J=1,NLOCT
          JBAS(I,J)=0
          IF(YBEST(I,J))JBAS(I,J)=1
 1110 CONTINUE
 1120 WRITE(6,02)(JBAS(I,J),J=1,NLOCT)
C
C
 1500 IF(MORE)2000,2000,10
C
 2000 STOP
C
C
    1 FORMAT(10X,15I8)
    2 FORMAT(10X,40I3)
    3 FORMAT(10X,'DATA FOR PROBLEM NO.',I5/
     *       10X,'   NO OF FACILITIES ',I5/
     *       10X,'   NO OF LOCATIONS  ',I5//)
    4 FORMAT(10X,'   VALUE OF LAMDA   ',I5/
     *       10X,'   TOL FOR CONVERGED ',I5/
     *       10X,'   UB ON ITERATIONS  ',I5//)
```

```
005215      000          EQUIVALENCE( MSTER(
005214      000          EQUIVALENCE( MSTER(
005215      000          EQUIVALENCE( MSTER(
005216      000          EQUIVALENCE( MSTER(
005217      000    C
005218      000          EQUIVALENCE( MSTER(
005219      000          EQUIVALENCE( MSTER(
005220      000          EQUIVALENCE( MSTER(
005221      000          EQUIVALENCE( MSTER(
005222      000    C
005223      000          EQUIVALENCE( MSTER(
005224      000          EQUIVALENCE( MSTER(
005225      000    C
005226      000    C                                                              GAP7 439
005227      000    C     IPDATE CALLS COUNTERS                                    GAP7 440
005228      000            KUVW=KUVW+1
005229      000            K1=30
005230      000    C                                                              GAP7 442
005231      000            IF(KSNSW(21).EQ.1.AND.K
005232      000    C                                                              GAP7 446
005233      000    C                                                              GAP7 447
005234      000            GO TO (130,782),KAH                                    GAP7 449
005235      000        130 NMIN1=NFCIL-1                                          GAP7 450
005236      000            NMIN2=NLOCT-1                                          GAP7 453
005237      000    C                                                              GAP7 454
005238      000    C                                                              GAP7 455
005239      000            DO 190 I=1,NFCIL                                       GAP7 456
005240      000            DO 165 K=1,NFCIL                                       GAP7 457
005241      000            WORK3(K)=F(I,K)                                        GAP7 458
005242      000            WORK4(K)=K                                             GAP7 459
005243      000        165 CONTINUE                                               GAP7 460
005244      000            DO 180 J=1,NMIN1                                       GAP7 461
005245      000            K=J+1                                                  GAP7 462
005246      000            DO 170 L=K,NFCIL                                       GAP7 463
005247      000            IF(WORK3(J).GT.WORK3(L)) GO TO 170                     GAP7 464
005248      000            KEEP=WORK3(J)                                          GAP7 465
005249      000            WORK3(J)=WORK3(L)                                      GAP7 466
005250      000            WORK3(L)=KEEP                                          GAP7 467
005251      000            KEEP=WORK4(J)                                          GAP7 468
005252      000            WORK4(J)=WORK4(L)                                      GAP7 469
005253      000            WORK4(L)=KEEP                                          GAP7 470
005254      000        170 CONTINUE                                               GAP7 471
005255      000        180 CONTINUE                                               GAP7 472
005256      000            DO 185 J=1,NFCIL                                       GAP7 473
005257      000        185 INTRI(I,J)=WORK4(J)                                    GAP7 474
005258      000            IF(KSNSW(23).EQ.1)WRITE(6,111)I,(WORK3(L),L=1,NFCIL)   GAP7 475
005259      000        190 CONTINUE                                               GAP7 476
005260      000    C                                                              GAP7 477
005261      000    C                                                              GAP7 478
005262      000    C                                                              GAP7 479
005263      000            DO 230 JJ=1,NLOCT                                      GAP7 480
005264      000            DO 192 K=1,NLOCT                                       GAP7 481
005265      000            WORK6(K)=D(JJ,K)                                       GAP7 482
005266      000            IF(WORK6(K).EQ.0)WORK6(K)=9999                         GAP7 483
005267      000            WORK4(K)=K                                             GAP7 484
005268      000        192 CONTINUE                                               GAP7 485
005269      000    C
```

```
005270  000          DO 210 J=1,NMIN2                                          QAP7 486
005271  000          K=J+1                                                     QAP7 487
005272  000          DO 200 L=K,NLOCT                                          QAP7 488
005273  000          IF(WORK6(J).LT.WORK6(L))GO TO 200                         QAP7 489
005274  000          KEEP=WORK6(J)                                             QAP7 490
005275  000          WORK6(J)=WORK6(L)                                         QAP7 491
005276  000          WORK6(L)=KEEP                                             QAP7 492
005277  000          KEEP=WORK4(J)                                             QAP7 493
005278  000          WORK4(J)=WORK4(L)                                         QAP7 494
005279  000          WORK4(L)=KEEP                                             QAP7 495
005280  000      200 CONTINUE                                                  QAP7 496
005281  000      210 CONTINUE                                                  QAP7 497
005282  000          IF(KSNSW(23).EQ.1)WRITE (6,12)JJ,(WORK6(L),L=1,NLOCT)     QAP7 498
005283  000          DO 220 K=1,NLOCT                                          QAP7 499
005284  000          WORK2(JJ,K)=WORK6(K)                                      QAP7 500
005285  000      220 DISTI(JJ,K)=WORK4(K)                                      QAP7 501
005286  000      230 CONTINUE                                                  QAP7 502
005287  000      C                                                             QAP7 503
005288  000      C                                                             QAP7 504
005289  000      C   CONSTRUCT THR WORK1 MATRIX                                QAP7 505
005290  000      222 DO 320 I=1,NFCIL                                          QAP7 506
005291  000          KEEP=0                                                    QAP7 508
005292  000          DO 250 L=1,NFCIL                                          QAP7 509
005293  000          LL=INTRI(I,L)                                             QAP7 510
005294  000          KEEP=KEEP+1                                               QAP7 512
005295  000          WORK3(KEEP)=F(I,LL)                                       QAP7 513
005296  000      250 CONTINUE                                                  QAP7 514
005297  000      C                                                             QAP7 516
005298  000          DO 290 J=1,NLOCT                                          QAP7 517
005299  000          YBEST(I,J)=.FALSE.
005300  000              Y(I,J)=.FALSE.
005301  000          KEEP=0                                                    QAP7 520
005302  000          DO 260 L=1,NLOCT                                          QAP7 521
005303  000          LL=DISTI(J,L)                                             QAP7 522
005304  000          KEEP=KEEP+1                                               QAP7 524
005305  000          WORK4(KEEP)=D(J,LL)                                       QAP7 525
005306  000      260 CONTINUE                                                  QAP7 526
005307  000          KEEP=0                                                    QAP7 527
005308  000      C   HERE OE CALCULATE OHAT IS EQUIVALENT TO AN OLD IC3 ELEMENT QAP7 529
005309  000          DO 270 L=1,NFCIL
005310  000      270 KEEP=KEEP+(WORK3(L)*WORK4(L))                             QAP7 531
005311  000      C                                                             QAP7 539
005312  000          WORK1(I,J)=KEEP
005313  000      290 CONTINUE                                                  QAP7 543
005314  000      C                                                             QAP7 545
005315  000      320 CONTINUE                                                  QAP7 550
005316  000      C                                                             QAP7 551
005317  000      C   CALCULATE THE REDUCTIONS                                  QAP7 552
005318  000          KEEP=0                                                    QAP7 553
005319  000          DO 350 I=1,NFCIL                                          QAP7 554
005320  000          WORK3(I)=32767                                            QAP7 555
005321  000          DO 330 J=1,NLOCT                                          QAP7 556
005322  000          IF(WORK1(I,J).GE.WORK3(I))GO TO 330                       QAP7 557
005323  000          WORK3(I)=WORK1(I,J)                                       QAP7 558
005324  000      330 CONTINUE                                                  QAP7 559
005325  000          KEEP=KEEP+WORK3(I)                                        QAP7 560
005326  000          DO 340 J=1,NLOCT                                          QAP7 561
```

```
000327      000          WORK2(I,J)=WORK1(I,J)                                                  GAP7 562
000328      000        340 WORK1(I,J)=WORK1(I,J)-WORK3(I)                                       GAP7 563
000329      000        350 CONTINUE                                                           GAP7 564
000330      000      C                                                                        GAP7 565
000331      000            DO 400 J=1,NLOCT                                                   GAP7 566
000332      000            WORK6(J)=32767                                                     GAP7 567
000333      000            DO 375 I=1,NFCIL                                                   GAP7 568
000334      000            IF(WORK1(I,J).GE.WORK6(J)) GO TO 375                               GAP7 569
000335      000            WORK6(J)=WORK1(I,J)                                                GAP7 570
000336      000        375 CONTINUE                                                           GAP7 571
000337      000            DO 380 I=1,NFCIL                                                   GAP7 572
000338      000        380 WORK1(I,J)=WORK1(I,J)-WORK6(J)                                     GAP7 573
000339      000        400 CONTINUE                                                           GAP7 574
000340      000            IF(NFCIL.EQ.NLOCT)GO TO 410                                        GAP7 575
000341      000            DO 405 I=1,10                                                      GAP7 576
000342      000        405 WRITE(6,98)                                                        GAP7 577
000343      000         98 FORMAT(02X,2(' ***** YOU HAVE TO ARRANGE AN                        GAP7 578
000344      000           * ARRAY IN LABCOM SO THAT THE SUBROUTINE                          GAP7 579
000345      000            STOP                                                               GAP7 580
000346      000        410 DO 415 J=1,NFCIL                                                   GAP7 581
000347      000        415 KEEP=KEEP+WORK6(J)                                                 GAP7 582
000348      000            LBCST=KEEP                                                         GAP7 583
000349      000      C                                                                        GAP7 584
000350      000      C   PRINT THE MATRICES IF YOU WISH                                       GAP7 585
000351      000            IF(KSNSW(24).EQ.0)GO TO 450                                        GAP7 586
000352      000            WORK4(1)=9999                                                      GAP7 587
000353      000            DO 420 J=2,40                                                      GAP7 588
000354      000        420 WORK4(J)=J-1                                                       GAP7 589
000355      000            WRITE(6,16)                                                        GAP7 590
000356      000            WRITE(6,01)(WORK4(J),J=1,40)                                       GAP7 591
000357      000            WRITE(6,07)                                                        GAP7 592
000358      000            DO 430 I=1,NFCIL                                                   GAP7 593
000359      000        430 WRITE(6,01)I,(WORK2(I,J),J=1,NLOCT),WORK3(I)                       GAP7 594
000360      000            WRITE(6,07)                                                        GAP7 595
000361      000            WRITE(6,01)NFCIL,(WORK6(J),J=1,NLOCT),KEEP                         GAP7 596
000362      000            WRITE(6,07)                                                        GAP7 597
000363      000            WRITE(6,18)                                                        GAP7 598
000364      000            WRITE(6,01)(WORK4(J),J=1,40)                                       GAP7 599
000365      000            WRITE(6,07)                                                        GAP7 600
000366      000            DO 440 I=1,NFCIL                                                   GAP7 601
000367      000        440 WRITE(6,01)I,(WORK1(I,J),J=1,NLOCT),WORK3(I)                       GAP7 602
000368      000            WRITE(6,07)                                                        GAP7 603
000369      000            WRITE(6,01)NFCIL,(WORK6(J),J=1,NLOCT),KEEP                         GAP7 604
000370      000            WRITE(6,07)                                                        GAP7 605
000371      000      C                                                                        GAP7 606
000372      000        450 IF(KSNSW(25).EQ.1)WRITE(6,19)KEEP                                  GAP7 607
000373      000      C                                                                        GAP7 608
000374      000            ITOT=0                                                             GAP7 613
000375      000      C                                                                        GAP7 614
000376      000      C     ASSIGN THE INITIAL VALUES OF THE U'S AND V'S                       GAP7 615
000377      000            K=0                                                                GAP7 616
000378      000            DO 480 I=1,NFCIL                                                   GAP7 617
000379      000            K=0                                                                GAP7 618
000380      000            DO 480 J=1,NLOCT                                                   GAP7 619
000381      000            K=K+1                                                              GAP7 620
000382      000            U(I,J)=-0*1000000                                                  GAP7 621
000383      000            V(I,J)=-0*1000000                                                  GAP7 622
```

```
005384    000          UBEST(I,J)=U(I,J)
005385    000          VBEST(I,J)=V(I,J)
005386    000        480 CONTINUE                                              QAP7 621
005387    000    C                                                             QAP7 622
005388    000    C   INITIALIZE HADY                                          QAP7 623
005389    000        750 N1=NFCIL                                              QAP7 624
005390    000            IF(NFCIL.EQ.NLOCT)GO TO 760                           QAP7 625
005391    000    C   ADD ADUMMY FACILITY                                       QAP7 626
005392    000            N1=NFCIL+1                                            QAP7 627
005393    000            DO 755 J=1,NLOCT                                      QAP7 628
005394    000        755 WORK1(N1,J)=0                                         QAP7 629
005395    000            IORIG(N1)=NLOCT-NFCIL                                 QAP7 630
005396    000        760 DO 765 I=1,NFCIL                                      QAP7 631
005397    000        765 IORIG(I)=1                                            QAP7 632
005398    000            DO 770 J=1,NLOCT                                      QAP7 633
005399    000        770 IDEST(J)=1                                            QAP7 634
005400    000            KEY=2                                                 QAP7 635
005401    000        773 CALL HADY(N1,K1,NLOCT,IORIG,IDEST,ITOT,KEY)
005402    000            GO TO(775,781),KEY                                    QAP7 637
005403    000        775 CALL GARY(N1,K1,NLOCT,IORIG,IDEST,ITOT)               QAP7 638
005404    000            GO TO(773,780),KEY                                    QAP7 639
005405    000        780 WRITE(6,99)
005406    000            STOP
005407    000         99 FORMAT(02X,3(' ***** ERROR IN GARY ***'))             QAP7 641
005408    000        781 MBCST=ITOT+LBCST                                      QAP7 644
005409    000            IF(KSNSW(28).EQ.1)WRITE(6,29)ITOT                     QAP7 645
005410    000    C                                                             QAP7 646
005411    000              IC2=0.5*FLOAT(MBCST)                                QAP7 647
005412    000            KEY=1                                                 QAP7 649
005413    000    C   EVALUATE THE SOLUTION
005414    000        782 DO 784 K=1,NLOCT                                      QAP7 651
005415    000        784 WORK8(K)=0
005416    000            LTOT=0                                                QAP7 652
005417    000            DO 790 I=1,NFCIL                                      QAP7 653
005418    000            DO 785 K=1,NLOCT                                      QAP7 654
005419    000            IF(WORK8(K).EQ.1)GO TO 785                            QAP7 655
005420    000            IF(JBAS(I,K).LE.K1)GO TO 785                          QAP7 656
005421    000            WORK4(I)=K
005422    000            Y(I,K)=.TRUE.
005423    000            IF(KAM.EQ.1)YBEST(I,K)=.TRUE.                         QAP7 657
005424    000            WORK8(K)=1                                            QAP7 658
005425    000            GO TO 790                                             QAP7 659
005426    000        785 CONTINUE                                              QAP7 660
005427    000        790 CONTINUE                                              QAP7 661
005428    000    C
005429    000            IF(KSNSW(30).EQ.0)GO TO 798
005430    000            DO 796 I=1,NFCIL
005431    000            DO 795 J=1,NLOCT
005432    000            WORK2(I,J)=0
005433    000            IF(Y(I,J))WORK2(I,J)=1
005434    000        795 CONTINUE
005435    000            WRITE(6,01)(WORK2(I,J),J=1,NLOCT)
005436    000        796 CONTINUE
005437    000        798 DO 810 I=1,NFCIL                                      QAP7 662
005438    000            DO 800 J=1,NFCIL                                      QAP7 663
005439    000            LOCI=WORK4(I)                                         QAP7 664
005440    000            LOCJ=WORK4(J)                                         QAP7 665
```

```
005441      000      800 LTOT=LTOT+(F(I,J)                                                    GAP7 666
005442      000      810 CONTINUE                                                             GAP7 667
005443      000          KTOT=0.5*FLOAT(LTOT)
005444      000    C                                                                          GAP7 670
005445      000          IF(KAM.EQ.2)RETURN
005446      000    C
005447      000          LBCST=IC2                                                            GAP7 680
005448      000    C                                                                          GAP7 681
005449      000          IF(KSNSW(26).EQ.1)                                                   GAP7 685
005450      000          IF(KSNSW(29).EQ.1)                                                   GAP7 686
005451      000    C                                                                          GAP7 687
005452      000          RETURN                                                               GAP7 688
005453      000    C                                                                          GAP7 689
005454      000        1 FORMAT(10X,20I4)                                                     GAP7 690
005455      000        6 FORMAT(1H1)                                                          GAP7 691
005456      000        7 FORMAT(/)                                                            GAP7 692
005457      000       11 FORMAT(/10X,'INTE                                                    GAP7 693
005458      000       12 FORMAT(/10X,'D                                                       GAP7 694
005459      000       14 FORMAT(1H+,110X,                                                     GAP7 695
005460      000       16 FORMAT(/02X,'MAT                                                     GAP7 696
005461      000          *       10X,'CO                                                      GAP7 697
005462      000       18 FORMAT(10X,'REQ                                                      GAP7 698
005463      000       19 FORMAT(100X,'AB                                                      GAP7 701
005464      000       23 FORMAT(/10X,'LO                                                      GAP7 704
005465      000       28 FORMAT(1H0,'                                                         GAP7 711
005466      000       36 FORMAT(90X,'VALU                                                     GAP7 712
005467      000       39 FORMAT(90X,'VALU                                                     GAP7 715
005468      000    C                                                                          GAP7 716
005469      000    C                                                                          GAP7 717
005470      000          END
005471      000    CFLI,SI TUNER=S*NSW0*                                                    060717
005472      000    C   A SUBROUTINE TO                                                      GAP71100
005473      000    C   THIS VERSION IS                                                      GAP71101
005474      000          SUBROUTINE DTA                                                    P,ESC,ISAVER,KAM)
005475      000          COMMON NFCIL,
005476      000          COMMON MSTR(                                                      ),KSNSW(60)
005477      000          COMMON NKNAP,KI
005478      000          INTEGER   F(20,                                                   TIX(25),VEC(20),WORK1(20,20)
005479      000          EQUIVALENCE( MS                                                   ))
005480      000          EQUIVALENCE( M                                                    ))
005481      000          EQUIVALENCE( MST                                                  (1))
005482      000          EQUIVALENCE( MST                                                  (1))
005483      000          EQUIVALENCE( IW                                                   (1,1))
005484      000    C                                                                          GAP71112
005485      000    C   READ THE NUMBER OF                                                   , THE NUMBER OF LOCATIONS , THE FRACT GAP71113
005486      000    C   OF THE INTERVAL (IT                                                 , THE NUMBER OF FIXED FACILITIES , AND  GAP71114
005487      000    C   THE CONTINUATION PA                                                 GAP71115
005488      000          READ(5,1)JOB,NFCIL,                                               ISAVER,LAMDA,MORE,ISTOP,ITERM,MEW
005489      000    C                                                                          GAP71117
005490      000          IF(ITOT.EQ.0)ITOT=                                                GAP71118
005491      000          DO 10 I=1,40                                                      GAP71119
005492      000          DO 10 J=1,40                                                      GAP71120
005493      000          F(I,J)=0                                                          GAP71121
005494      000       10 D(I,J)=0                                                          GAP71122
005495      000    C                                                                          GAP71123
005496      000          DO 20 I=1,NFCIL                                                   GAP71124
005497      000       20 READ(5,2)(WORK1(I,J),J=1,NFCIL)                                   GAP71125
```

```
005498  000   C                                                                 QAP71126
005499  000         DO 30 I=1,NFCIL                                             QAP71127
005500  000         DO 25 J=1,NFCIL                                             QAP71128
005501  000         IF(I.EQ.J)GO TO 25                                          QAP71129
005502  000         IF(I.GT.J)F(I,J)=WORK1(I,J)                                 QAP71130
005503  000         IF(I.LT.J)D(I,J)=WORK1(I,J)                                 QAP71131
005504  000      25 CONTINUE                                                    QAP71132
005505  000      30 CONTINUE                                                    QAP71133
005506  000   C                                                                 QAP71134
005507  000         DO 40 I=1,NFCIL                                             QAP71135
005508  000         DO 40 J=1,NFCIL                                             QAP71136
005509  000         F(I,J)=F(J,I)                                               QAP71137
005510  000         D(J,I)=D(I,J)                                               QAP71138
005511  000      40 CONTINUE                                                    QAP71139
005512  000         DO 50 I=1,30
005513  000      50 KSNSW(I)=1
005514  000   C                                                                 QAP71140
005515  000   C                                                                 QAP71141
005516  000         KSNSW(1)=1                                                  QAP71142
005517  000         KSNSW(4)=1                                                  QAP71143
005518  000         KSNSW(6)=0                                                  QAP71144
005519  000         KSNSW(07)=0                                                 QAP71145
005520  000         KSNSW(16)=0
005521  000         KSNSW(14)=0
005522  000         KSNSW(08)=1                                                 QAP71146
005523  000         KSNSW(30)=0
005524  000         KSNSW(25)=0
005525  000         KSNSW(02)=0
005526  000         KSNSW(23)=0
005527  000         KSNSW(24)=0
005528  000   C   READ THE INDICES OF THE FACILITIES FIXED APRIORI               QAP71147
005529  000         DO 70 I=1,NFIXD                                             QAP71148
005530  000      70 READ(5,1)  FIX(I),WHERE(I)                                  QAP71149
005531  000   C                                                                 QAP71150
005532  000         NFIXD=1                                                     QAP71151
005533  000         RETURN                                                      QAP71152
005534  000   C                                                                 QAP71153
005535  000   C                                                                 QAP71154
005536  000   C                                                                 QAP71155
005537  000         1 FORMAT(20X,12I5)                                          QAP71156
005538  000         2 FORMAT(10X,35I2)                                          QAP71157
005539  000         END
005540  000   @ELT,SI TUMER-S*NEWQAP.KNAP,,,1377750860112
005541  000   C   A SUBROUTINE TO SOLVE THE KNAPSACK PROBLEM
005542  000         SUBROUTINE KNAP(I,K,J[illegible],KAM)
005543  000         COMMON NFCIL,NLOCI,LO[illegible]
005544  000         COMMON MSTER(05200),I[illegible],KSNSW(0060)
005545  000         COMMON NKNAP,KHADY,KOARY,NECTP
005546  000   C
005547  000   C
005548  000         INTEGER U(20,20),V(20,20)
005549  000         INTEGER F(20,20),D(20,20)
005550  000         EQUIVALENCE ( MSTER(0301),    F(1,1))
005551  000         EQUIVALENCE ( MSTER(0401),    D(1,1))
005552  000   C
005553  000         EQUIVALENCE (MSTER(0001),    U(1,1))
005554  000         EQUIVALENCE (MSTER(1201),    V(1,1))
```

```
005555      000    C
005556      000    C   UPDATE THE COUNTER BY 1
005557      000          NKNAP=NKNAP+1
005558      000          KAM=32767n000
005559      000          DO 130 J=1,NLOCT
005560      000          DO 120 L=1,NLOCT
005561      000          IF(J.EQ.L)GO TO 120
005562      000          KOMP=(1000000*F(I,K)+C(I,K))+U(I,J)+V(K,L)
005563      000          IF(KSNSW(06).EQ.1.AND.I.LE.2.AND.K.LE.2)WRITE(6,11)I,J,K,L,KOMP
005564      000       11 FORMAT(30X,'C(',4I3,')=',I7)
005565      000          IF(KOMP.GT.KAM)GO TO 120
005566      000           KAM=KOMP
005567      000          JEEP=J
005568      000          LEEP=L
005569      000    C
005570      000      120 CONTINUE
005571      000      130 CONTINUE
005572      000    C
005573      000          RETURN
005574      000    C
005575      000          END
005576      000    @FLT,SI TUMER=S*NEWOAP,[illegible]0053612
005577      000    C   ****8*********[illegible]*********[illegible]*********[illegible]
```

```
005612      000          NORG1=NORIG                                                    QAP7 752
005613      000          NCOL=1                                                         QAP7 753
005614      000          NROW=1                                                         QAP7 754
005615      000          ISAV=K3                                                        QAP7 755
005616      000          DO 130 I=1,NORIG                                               QAP7 756
005617      000          DO 130 J=1,NDEST                                               QAP7 757
005618      000        130 JBAS(I,J)=0                                                  QAP7 758
005619      000          DO 140 J=1,NDEST                                               QAP7 759
005620      000        140 IV(J)=1                                                      QAP7 760
005621      000      C   FROM HERE UNTIL JUST BEFORE 300 THE VARIABLES ARE ALLOCATED    QAP7 761
005622      000        150 IF(ICOST(NROW,NCOL)-ISAV)180,160,160                       QAP7 762
005623      000        160 NCOL=NCOL+1                                                QAP7 763
005624      000            IF(NCOL-NDEST)150,150,200                                  QAP7 764
005625      000        180 IF(IV(NCOL)-1)160,190,160                                  QAP7 765
005626      000        190 ISAV=ICOST(NROW,NCOL)                                      QAP7 766
005627      000            NCOL1=NCOL                                                 QAP7 767
005628      000            GO TO 160                                                  QAP7 768
005629      000      C                                                                QAP7 769
005630      000        200 IF(IORIG(NROW)-IDEST(NCOL1))210,230,310                    QAP7 770
005631      000        210 ISET=IORIG(NROW)+K1                                        QAP7 771
005632      000      C                                                                QAP7 772
005633      000            JBAS(NROW,NCOL1)=ISET                                      QAP7 773
005634      000      C                                                                QAP7 774
005635      000            IDEST(NCOL1)=IDEST(NCOL1)-IORIG(NROW)                      QAP7 775
005636      000            GO TO 290                                                  QAP7 776
005637      000      C                                                                QAP7 777
005638      000        230 ISET=IORIG(NROW)+K1                                        QAP7 778
005639      000      C                                                                QAP7 779
005640      000            JBAS(NROW,NCOL1)=ISET                                      QAP7 780
005641      000      C                                                                QAP7 781
005642      000            IV(NCOL1)=0                                                QAP7 782
005643      000      C   MAKE NECESSARY ARRANGEMENTS FOR DEGENERACY                    QAP7 783
005644      000            NCOL=1                                                     QAP7 784
005645      000            ISAV=K3                                                    QAP7 785
005646      000            IF(NROW-NORIG)240,320,320                                  QAP7 786
005647      000        240 IF(ICOST(NROW,NCOL)-ISAV)260,270,270                       QAP7 787
005648      000        250 IF(IV(NCOL)-1)270,260,270                                  QAP7 788
005649      000        260 ISAV=ICOST(NROW,NCOL)                                      QAP7 789
005650      000            NCOL1=NCOL                                                 QAP7 790
005651      000        270 NCOL=NCOL+1                                                QAP7 791
005652      000            IF(NCOL-NDEST)240,240,280                                  QAP7 792
005653      000      C                                                                QAP7 793
005654      000        280 JBAS(NROW,NCOL1)=K1                                        QAP7 794
005655      000      C                                                                QAP7 795
005656      000        290 NROW=NROW+1                                                QAP7 796
005657      000            IF(NROW-NORIG)300,300,320                                  QAP7 797
005658      000        300 NCOL=1                                                     QAP7 798
005659      000            ISAV=K3                                                    QAP7 799
005660      000            GO TO 150                                                  QAP7 800
005661      000      C                                                                QAP7 801
005662      000        310 ISET=IDEST(NCOL1)+K1                                       QAP7 802
005663      000      C                                                                QAP7 803
005664      000            JBAS(NROW,NCOL1)=ISET                                      QAP7 804
005665      000      C                                                                QAP7 805
005666      000            IORIG(NROW)=IORIG(NROW)-IDEST(NCOL1)                       QAP7 806
005667      000            IV(NCOL1)=0                                                QAP7 807
005668      000            GO TO 300                                                  QAP7 808
```

```
005669      000    C   <<++++++
```

```
005726      000      640 ISAV=IS1                                                                  QAP7 866
005727      000    C                                                                               QAP7 867
005728      000          NROW1=NROW                                                                QAP7 868
005729      000          NCOL1=NCOL                                                                QAP7 869
005730      000          GO TO 570                                                                 QAP7 870
005731      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 871
005732      000    C   CURRENT SOLUTION IS NOT OPTIMAL                                           QAP7 872
005733      000    C   ------------------------------------------------------------------------- QAP7 873
005734      000      700 KEY=1                                                                     QAP7 874
005735      000          RETURN                                                                    QAP7 875
005736      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 876
005737      000    C   CURRENT SOLUTION IS OPTIMAL                                               QAP7 877
005738      000    C   ------------------------------------------------------------------------- QAP7 878
005739      000      800 KEY=2                                                                     QAP7 879
005740      000    C   CALCULATE ACTUAL VALUES OF BASIC VARIABLES                                 QAP7 880
005741      000          ITOT=0                                                                    QAP7 881
005742      000          DO 850 I=1,NORIG                                                          QAP7 882
005743      000          DO 830 J=1,NDEST                                                          QAP7 883
005744      000          IF(JBAS(I,J))830,830,820                                                  QAP7 884
005745      000      820 JBASE    =JBAS(I,J)-M1                                                    QAP7 885
005746      000          ITOT=ITOT+(JBASE    *ICOST(I,J))                                         QAP7 886
005747      000      830 CONTINUE                                                                  QAP7 887
005748      000      850 CONTINUE                                                                  QAP7 888
005749      000          IF(KSNSW(30).EQ.0)GO TO 900                                               QAP7 889
005750      000          WRITE(6,11)                                                               QAP7 890
005751      000          DO 870 I=1,NORIG                                                          QAP7 891
005752      000      870 WRITE(6,01)I,(JBAS(I,J),J=1,NDEST)                                        QAP7 892
005753      000    C                                                                               QAP7 893
005754      000      900 RETURN                                                                    QAP7 894
005755      000    C                                                                               QAP7 895
005756      000        1 FORMAT(10X,I3,2X,2013)                                                    QAP7 896
005757      000       11 FORMAT(/10X,'OPTIMAL SOLUTION OF THE LINEAR ASSIGNMENT PROBLEM '/)        QAP7 897
005758      000    C                                                                               QAP7 898
005759      000    C                                                                               QAP7 899
005760      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 900
005761      000          END                                                                       QAP7 901
005762      000    @FLT,SI TUMER-S*NEWQAP.GARY,,,1@@@@3053612
005763      000    C                                                                               QAP7 902
005764      000    C                                                                               QAP7 903
005765      000    C                                                                               QAP7 904
005766      000    C   ************************************************************************** QAP7 905
005767      000    C   A SUBROUTINE TO IDENTIFY THE LOOP AND UPDATE ITS ENTRIES                   QAP7 906
005768      000    C   <<++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 907
005769      000          SUBROUTINE GARY(NORIG,M1,NDEST,NCOL1,KEY)                                 QAP7 908
005770      000    C   ------------------------------------------------------------------------- QAP7 909
005771      000          COMMON NFCIL,NDEST,LOST,ITOT                                              QAP7 910
005772      000          COMMON MSTER(05200),IWORK(1000),KSNSW(60)                                 QAP7 911
005773      000          COMMON NKNAP,KHADY,KGARY,NECTP                                            QAP7 912
005774      000    C                                                                               QAP7 913
005775      000          DIMENSION JBAS(20,20)
005776      000          DIMENSION  INET(080),INET1(20),INET2(20)
005777      000    C                                                                               QAP7 916
005778      000          EQUIVALENCE( IWORK(0001),INET1(001))                                      QAP7 917
005779      000          EQUIVALENCE( IWORK(0061),INET2(001))                                      QAP7 918
005780      000          EQUIVALENCE( IWORK(0081), INET(001))                                      QAP7 919
005781      000          EQUIVALENCE( IWORK(1001), JBAS(1,1))                                      QAP7 920
005782      000    C                                                                               QAP7 921
```

```
005785      000    C                                                                    GAP7 922
005784      000    C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++GAP7 923
005785      000    C                                                                    GAP7 924
005786      000    C                                                                    GAP7 925
005787      000    C   UPDATE CALLS COUNTER                                            GAP7 926
005788      000          KGARY=KGARY+1                                                  GAP7 927
005789      000    C                                                                    GAP7 928
005790      000    C   A) INITIALZE                                                     GAP7 929
005791      000    C    * SET K2 .GE. (10*K1)                                          GAP7 930
005792      000          K2=300000                                                      GAP7 931
005793      000          K=NORIG+NDEST                                                  GAP7 932
005794      000          K=2*K                                                          GAP7 933
005795      000          DO 20 I=1,K                                                    GAP7 934
005796      000       20 INET(I)=0                                                      GAP7 935
005797      000    C    ** THUS THE SIZE OF ARRAY INET .GE. 2*(NORIG+NDEST+1) **       GAP7 936
005798      000          DO 40 I=1,NORIG                                                GAP7 937
005799      000       40 INET1(I)=0                                                     GAP7 938
005800      000          DO 60 I=1,NDEST                                                GAP7 939
005801      000       60 INET2(I)=0                                                     GAP7 940
005802      000    C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++GAP7 941
005803      000    C   B) START SEARCHING FOR [illegible] LOOP                         GAP7 942
005804      000    C   ---------------------------------------------------------------GAP7 943
005805      000    C      ** FROM HERE UNTIL [illegible] 900 GAP [illegible] ARE       GAP7 944
005806      000    C         1) RECOGNIZING [illegible] OF LOOP [illegible] ITS ROW AND GAP7 945
005807      000    C            COLUMN NE[illegible] [illegible] JACENT L[illegible] OF ARRAY INET GAP7 946
005808      000    C         2) DETECT ANY [illegible] ERRORS                          GAP7 947
005809      000    C   ---------------------------------------------------------------GAP7 948
005810      000          I=1                                                            GAP7 949
005811      000          INET(I)=NROW1                                                  GAP7 950
005812      000          I1=I+1                                                         GAP7 951
005813      000          INET(I1)=NCOL1                                                 GAP7 952
005814      000          NROW=NROW1                                                     GAP7 953
005815      000          NCOL=1                                                         GAP7 954
005816      000          I=I+2                                                          GAP7 955
005817      000      100 IF(JBAS(NROW,NCOL)-K1)[illegible]                            GAP7 956
005818      000      120 IF(NCOL-NCOL1)140,[illegible]                                GAP7 957
005819      000      140 INET(I)=NROW                                                 GAP7 958
005820      000          I1=I+1                                                         GAP7 959
005821      000          INET(I1)=NCOL                                                  GAP7 960
005822      000          I=I+2                                                          GAP7 961
005823      000          GO TO 220                                                      GAP7 962
005824      000    C                                                                    GAP7 963
005825      000      160 NCOL=NCOL+1                                                  GAP7 964
005826      000          IF(NCOL-NDEST)100,100,[illegible]                            GAP7 965
005827      000      200 I=1                                                          GAP7 966
005828      000    C     *) NO BASIS ELEMENT IN THE ROW OF THE ENTERING , ERROR TYPE 1 GAP7 967
005829      000          WRITE(6,1)I,NROW1,NCOL1                                        GAP7 968
005830      000          KEY=2                                                          GAP7 969
005831      000    C                                                                    GAP7 970
005832      000          RETURN                                                         GAP7 971
005833      000    C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++GAP7 972
005834      000      220 INET2(NCOL)=1                                                GAP7 973
005835      000          NROW=1                                                         GAP7 974
005836      000      240 IF(JBAS(NROW,NCOL)-K1)[illegible]                            GAP7 975
005837      000      260 NROW=NROW+1                                                  GAP7 976
005838      000          IF(NROW-NORIG)240,240,[illegible]                            GAP7 977
005839      000      300 I=I-2                                                        GAP7 978
```

```
005840  000          IF(I)360,360,340                                                  QAP7 979
005841  000    340 NROW=INET(I)                                                        QAP7 980
005842  000          I1=I+1                                                            QAP7 981
005843  000          NCOL=INET(I1)                                                     QAP7 982
005844  000          INET2(NCOL)=0                                                     QAP7 983
005845  000    C                                                                       QAP7 984
005846  000          GO TO 520                                                         QAP7 985
005847  000    C   *) THERE IS NO BASIS LOOP                                          QAP7 986
005848  000    360 I=2                                                                 QAP7 987
005849  000          WRITE(6,2)I,NROW1,NCOL1                                           QAP7 989
005850  000          KEY=2                                                             QAP7 990
005851  000    C                                                                       QAP7 991
005852  000          RETURN                                                            QAP7 992
005853  000    C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP7 993
005854  000    380 I2=I-2                                                              QAP7 994
005855  000          IF(NROW-INET(I2))400,[illegible],400                              QAP7 995
005856  000    400 IF(INET1(NROW))300,420,[illegible]                                  QAP7 996
005857  000    420 INET(I)=NROW                                                        QAP7 997
005858  000          I1=I+1                                                            QAP7 998
005859  000          INET(I1)=NCOL                                                     QAP7 999
005860  000          I=I+2                                                             QAP71000
005861  000          IF(NROW-NROW1)480,460,[illegible]                                 QAP71001
005862  000    460 I=I-2                                                               QAP71002
005863  000          GO TO 300                                                         QAP71003
005864  000    480 INET1(NROW)=1                                                       QAP71004
005865  000          NCOL=1                                                            QAP71005
005866  000    500 IF(JBAS(NROW,NCOL)-K1)520,[illegible],560                           QAP71006
005867  000    520 NCOL=NCOL+1                                                         QAP71007
005868  000          IF(NCOL-NDEST)500,500,660                                         QAP71008
005869  000    540 I1=I-1                                                              QAP71009
005870  000          IF(NCOL-INET(I1))580,[illegible],560                              QAP71010
005871  000    560 IF(INET2(NCOL))640,600,[illegible]                                  QAP71011
005872  000    600 INET(I)=NROW                                                        QAP71012
005873  000          I1=I+1                                                            QAP71013
005874  000          INET(I1)=NCOL                                                     QAP71014
005875  000          I=I+2                                                             QAP71015
005876  000          IF(NCOL-NCOL1)220,700,220                                         QAP71016
005877  000    640 I=I-2                                                               QAP71017
005878  000          IF(I)360,360,680                                                  QAP71018
005879  000    680 NROW=INET(I)                                                        QAP71019
005880  000          I1=I+1                                                            QAP71020
005881  000          NCOL=INET(I1)                                                     QAP71021
005882  000          INET1(NROW)=0                                                     QAP71022
005883  000          GO TO 260                                                         QAP71023
005884  000    C                                                                       QAP71024
005885  000    700 I=3                                                                 QAP71025
005886  000    C   *) CALCULATE THE MINIMUM ENTRY IN BASIS LOOP                       QAP71026
005887  000          ISAV=K2                                                           QAP71027
005888  000    720 NROW=INET(I)                                                        QAP71028
005889  000          I1=I+1                                                            QAP71029
005890  000          NCOL=INET(I1)                                                     QAP71030
005891  000          IF(JBAS(NROW,NCOL)-ISAV)760,780,780                               QAP71031
005892  000    760 ISAV=JBAS(NROW,NCOL)                                                QAP71032
005893  000          NROW2=NROW                                                        QAP71033
005894  000          NCOL2=NCOL                                                        QAP71034
005895  000    780 IF(NCOL-NCOL1)800,860,800                                           QAP71035
005896  000    800 I=I+4                                                               QAP71036
```

```
      IF(I-K)720,720,800                                                    GAP71037
C   *) THE BASIS LOOP HAS MORE ENTRIES THAN THERE ARE IN THE BASIS ,        GAP71038
C      ERROR TYPE 3                                                         GAP71039
  840 I=3                                                                   GAP71040
C                                                                           GAP71041
      WRITE(6,3)I,NROW1,NCOL1                                               GAP71042
      KEY=2                                                                 GAP71043
C                                                                           GAP71044
      RETURN                                                                GAP71045
C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++GAP71046
  860 IF(ISAV-K2)900,860,840                                                GAP71047
C   *) THERE IS NO ELEMENT IN THE BASIS LOOP (I, K2, ERROR TYPE 4           GAP71048
  860 I=4                                                                   GAP71049
C                                                                           GAP71050
      WRITE(6,4)I,NROW1,NCOL1                                               GAP71051
      KEY=2                                                                 GAP71052
C                                                                           GAP71053
      RETURN                                                                GAP71054
C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++GAP71055
C   C) NOW THE LOCATIONS OF ALL ELEMENTS IN THE BASIS ARE SEQUENTIALLY      GAP71056
C      ARRANGED (FAIRWISE) IN A WAY THAT AND EVERY ROW CONTAINS THE MINIMUM GAP71057
C      ENTRY IN THE LOWER PART OF THE UPDATED MATRIX. SOLUTION BY           GAP71058
C      SUCCESSIVE ADDITIONS AND SUBTRACTIONS OVER THE ELEMENTS OF THE       GAP71059
C      BASIS LOOP                                                           GAP71060
C   ---------------------------------------------------------------------GAP71061
  900 J=-1                                                                  GAP71062
      NROW=INET(1)                                                          GAP71063
      NCOL=INET(2)                                                          GAP71064
      JBAS(NROW,NCOL)=ISAV                                                  GAP71065
      ISAV=ISAV-K1                                                          GAP71066
C                                                                           GAP71067
      I=3                                                                   GAP71068
  920 NROW=INET(I)                                                          GAP71069
      I1=I+1                                                                GAP71070
      NCOL=INET(I1)                                                         GAP71071
      IF(NROW-NROW2)1000,940,1000                                           GAP71072
  940 IF(NCOL-NCOL2)1000,960,1000                                           GAP71073
  960 JBAS(NROW,NCOL)=0                                                     GAP71074
      GO TO 1020                                                            GAP71075
C                                                                           GAP71076
 1000 ISET=JBAS(NROW,NCOL)                                                  GAP71077
      JBAS(NROW,NCOL)=ISAV                                                  GAP71078
C                                                                           GAP71079
 1020 J=-J                                                                  GAP71080
      I=I+2                                                                 GAP71081
      IF(NCOL-NCOL1)920,980,1040                                            GAP71082
C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++GAP71083
C   NOW THAT THE NEW BASIS SOLUTION HAS BEEN OBTAINED , SET KEY = 1 TO      GAP71084
C   ENABLE HADY TO CHECK THE KEY                                            GAP71085
C   ---------------------------------------------------------------------GAP71086
 1040 KEY=1                                                                 GAP71087
      RETURN                                                                GAP71088
C                                                                           GAP71089
    1 FORMAT(10X,'ERROR TYPE 1'/15X,'NO BASIS ELEMENT IN ROW WHICH CONTA    GAP71090
     *INS ENTERING ELEMENT IN THE ITERATION',I3)                            GAP71091
    2 FORMAT(10X,'ERROR TYPE 2'/15X,'THERE IS NO ELEMENT IN THE BASIS LOOP',3I3)  GAP71092
    3 FORMAT(10X,'ERROR TYPE 3'/15X,'THE BASIS LOOP HAS MORE ELEMENTS TH    GAP71093
```

```
005954      000        *AN THERE ARE IN THE BASIS',3I3)                                          QAP71094
005955      000       4 FORMAT(10X,'ERROR TYPE 4:  I =',I3,'THERE IS NO ELEMENT IN THE BASIS LQAP71095
005956      000        *OOP LESS THSN K2',3I3)                                                  QAP71096
005957      000   C                                                                             QAP71097
005958      000   C   <<+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++QAP71098
005959      000         END                                                                     QAP71099
005960      000   @ELT,SI TUMER-S*NEWQAP.X,,,130110061312
005961      000   C    A FUNCTION TO CALCULATE THE VALUE OF X
005962      000         FUNCTION X(I,J,K,L)
005963      000         COMMON NFCIL,NLOCT,
005964      000         COMMON MSTER(05200),KSNSW(0060)
005965      000         COMMON NKNAP,KHADY,
005966      000   C
005967      000         INTEGER INDEX(0400),KNDEX(0400),JNDEX(0400),LNDEX(0400)
005968      000   C
005969      000   C
005970      000         EQUIVALENCE (MSTER(
005971      000         EQUIVALENCE (MSTER(
005972      000         EQUIVALENCE (MSTER(
005973      000         EQUIVALENCE (MSTER(
005974      000   C
005975      000   C
005976      000         X=0
005977      000         KAM=0
005978      000      20 KAM=KAM+1
005979      000         IF(INDEX(KAM).EQ.99)RETURN
005980      000         IF(I.EQ.INDEX(KAM).AND.J.EQ.JNDEX(KAM))GO TO 50
005981      000         GO TO 20
005982      000      50 IF(K.EQ.KNDEX(KAM).AND.L.EQ.LNDEX(KAM))GO TO 90
005983      000         GO TO 20
005984      000   C
005985      000      90 X=1
005986      000   C
005987      000         RETURN
005988      000   C
005989      000         END
005990      000   @ELT,SI TUMER-S*QAP-DATA.PROBLEM1,,,130110061312
005991      000   PROBLEM              4001    5    5   70   01   -2   05 9999
005992      000   1      1   0001010203
005993      000   1      2   0500020102
005994      000   1      3   0203000102
005995      000   1      4   0400000001
005996      000   1      5   0102000500
005997      000                          1    1
005998      000   @ELT,SI TUMER-S*QAP-DATA.PROBLEM2,,,130262061312
005999      000   PROBLEM              4002    6    6   70   01   -2    8 9999    6
006000      000   2      1   000102010203
006001      000   2      2   050001020102
006002      000   2      3   020300030201
006003      000   2      4   040000000102
006004      000   2      5   010200050001
006005      000   2      6   000200021000
006006      000                          5    2
006007      000   @ELT,SI TUMER-S*QAP-DATA.PROBLEM3,,,120420061312
006008      000   PROBLEM              4003    7    7   70    2   -2   10 9999
006009      000   3      1   00010203020304
006010      000   3      2   05000102010203
```

```
000011      000   3      3    02030001070102
000012      000   3      4    04000100030703
000013      000   3      5    01020005000102
000014      000   3      6    00020202100001
000015      000   3      7    00020302000500
000016      000   4003  FIX       1          6      5
000017      000   4003  FIX       2          3
000018      000   @FLT-SI TUMER-S*QAP-DATA                                  1006140  1312
000019      000   PROBLEM                4004     3     4    70     3    -2    20 9999
000020      000   4      1    00010203010
000021      000   4      2    05000102020
000022      000   4      3    02030001030
000023      000   4      7    04000000030
000024      000   4      5    01020005000
000025      000   4      6    00020002100
000026      000   4      7    00020002000
000027      000   4      8    06000510000
000028      000   4004 FIX       1          7      2
000029      000   4004 FIX       2          8      1
000030      000   4004 FIX       3          6      1
000031      000   @FLT-SI TUMER-S*QAP-DATA                                  1000060  1312
000032      000   4003 CONFIGURATIONS  4003     1     3    70     3    -2    5552750  247  275
000033      000   5      1    00910203010
000034      000   5      2    05000102030
000035      000   5      3    02030001030
000036      000   5      4    04000100030
000037      000   5      5    01020005000
000038      000   5      6    00020002100
000039      000   5      7    00020002000
000040      000   5      8    06000510000
000041      000   5      9    02040500000
000042      000   5     10    01050200050
000043      000   5     11    01000203010
000044      000   5     12    01000205010
000045      000   FIXED    1             7      2
000046      000   FIXED    2             9      1
000047      000   FIXED    2            11      1
000048      000   @FLT-SI TUMER-S*QAP-DATA                                  1524770  1312
000049      000   PROBLEM                4003     1     3    60     5    -2 0100 0000  482  535
000050      000   6      1    00010203040
000051      000   6      2    10000102030
000052      000   6      3    00010001020
000053      000   6      4    05031000010
000054      000   6      5    01020201000
000055      000   6      6    00020001030
000056      000   6      7    01020205050
000057      000   6      8    02030500050
000058      000   6      9    02020400050
000059      000   6     10    02000502010
000060      000   6     11    02020201000
000061      000   6     12    00000200030
000062      000   6     13    04100502000
000063      000   6     14    00050505050
000064      000   6     15    00000500051
000065      000   4006 FIX       1         13      3
000066      000   4006 FIX       2          1      1
000067      000   4006 FIX       3         06     10
```

```
006066      000   4000 FIX      4          7    15
006069      000   4000 FIX      5         11    19
006070      000   @ELT,SI TUMER-S*QAP-DATA.PROBLEM,,,,115776061312
006071      000   PROBLEM                4007    20    20    75    10     0   300 8000
006072      000   7      1   0001020304010203[illegible]0405060304050607
006073      000   7      2   0000010203020105[illegible]0405040305060
006074      000   7      3   0503000102030201[illegible]0504050406
006075      000   7      4   0010020001040305[illegible]0306050403
006076      000   7      5   0505000100050405[illegible]0706050403
006077      000   7      6   0201050005000105[illegible]0203050605
006078      000   7      7   1005020506050001[illegible]0403020305
006079      000   7      8   0301040205020000[illegible]0403020304
006080      000   7      9   0102040102010001[illegible]0205040302
006081      000   7     10   0504050005060001[illegible]0605040302
006082      000   7     11   0502001002000510[illegible]0102030405
006083      000   7     12   0505000200001010[illegible]0302010203
006084      000   7     13   0000000205100[illegible]0203020102
006085      000   7     14   0010050001000[illegible]0104050201
006086      000   7     15   0510010201020510[illegible]0504050201
006087      000   7     16   0403000101000102[illegible]0001020304
006088      000   7     17   0400000505010405[illegible]0001020304
006089      000   7     18   0005050202000105[illegible]0005000102
006090      000   7     19   0010000505010002[illegible]0020201006
006091      000   7     20   0105000501051010[illegible]0000001060
006092      000                          15    15
006093      000                          01    01
006094      000                          02    03
006095      000                          04    05
006096      000                          05    07
006097      000                          09    11
006098      000                          11    13
006099      000                          13    01
006100      000                          18    10
006101      000                          20    12
006102      000   @ELT,SI TUMER-S*QAP-DATA.PROBLEM,,,,1206000061212
006103      000   PROBLEM                4008    30     0    70     0     0    10 9999
006104      000   8      1   0001020304050102[illegible]0304050607[illegible]0304050607080809
006105      000   8      2   0300010203040201[illegible]0304050607[illegible]0708090506060708
006106      000   8      3   0204000102030302[illegible]0203040506[illegible]0506050605040506
006107      000   8      4   0000030001020403[illegible]0302030405[illegible]0806070605040506
006108      000   8      5   0010040000010506[illegible]0403020307[illegible]0806070605040506
006109      000   8      6   0204000005000605[illegible]0403020307[illegible]0509050706050404
006110      000   8      7   1000050002010001[illegible]0304050607[illegible]0703040506060708
006111      000   8      8   0500050200021000[illegible]0102030405[illegible]0506040304050607
006112      000   8      9   0002050200021001[illegible]0102030405[illegible]0705050405040506
006113      000   8     10   0502010000010503[illegible]0302010207[illegible]0304050604030405
006114      000   8    111   0201040600041005[illegible]0302010206[illegible]0203070605040304
006115      000   8     12   0500010002101000[illegible]0403020107[illegible]0302080706050403
006116      000   8     13   0005000200100600[illegible]0102030405[illegible]0506020304050607
006117      000   8     14   0000040500020000[illegible]0102030402[illegible]0405030203040506
006118      000   8     15   0200000200050002[illegible]0102030405[illegible]0304040302030405
006119      000   8     16   0000040500051004[illegible]0401040001[illegible]0203050403020304
006120      000   8     17   0500000102000205[illegible]0005000061[illegible]0102060504030203
006121      000   8     18   0602060101050102[illegible]0301020006[illegible]0201070605040302
006122      000   8     19   0300030100001010[illegible]0000020500[illegible]0405010203040506
006123      000   8     20   0001020100000106[illegible]0001020001[illegible]0304020102030405
006124      000   8     21   0106050202000500[illegible]0002000205[illegible]0203030201020304
```

```
006125      000   8    22   1001050200100505[illegible]
006126      000   8    23   0000020405000205[illegible]
006127      000   8    24   1001010001000302[illegible]
006128      000   8    25   0202000200000505[illegible]
006129      000   8    26   0102000002040000[illegible]
006130      000   8    27   0105030201000405[illegible]
006131      000   8    28   0101010200100005[illegible]
006132      000   8    29   0010000502010100[illegible]
006133      000   8    30   0105020501010302[illegible]
006134      000
006135      000
006136      000   @ELT,SI TUMER-S*FATHOM.COPY,,,1341200051012
006137      000           SUBROUTINE COPY(N,A,B)
006138      000   C*        COPY COPIES ONE ARRAY INTO ANOTHER
006139      000           PARAMETER MAXPTS=80
006140      000           DIMENSION A(MAXPTS,MAXPTS),B(MAXPTS,MAXPTS)
006141      000           DO 10 I=1,N
006142      000           DO 10 J=1,N
006143      000        10 B(I,J)=A(I,J)
006144      000           RETURN
006145      000           END
006146      000   @ELT,SI TUMER-S*FATHOM.INPUT,,,1341[illegible]052113
006147      000           SUBROUTINE INPUT(NPTS,IFIN,COST,PRINTM)
006148      000   C*          INPUT READS UPPER TRIANGLE OF SYMMETRIC COST MATRIX
006149      000           PARAMETER MAXPTS=80
006150      000           DIMENSION COST(MAXPTS,MAXPTS)
006151      000           LOGICAL PRINTM
006152      000           IF(IFIN.NE.0)GO TO 20
006153      000   C*          READ MATRIX
006154      000           N=NPTS-1
006155      000           DO 10 I=1,N
006156      000           L=I+1
006157      000           READ(5,1000) (COST(I,J),J=L,NPTS)
006158      000    1000   FORMAT()
006159      000           COST(I,I)=1.0E38
006160      000           DO 10 J=L,NPTS
006161      000        10 COST(J,I)=COST(I,J)
006162      000           COST(NPTS,NPTS)=1.0E38
006163      000           GO TO 30
006164      000   C*        USE MATRIX PREVIOUSLY SAVED
006165      000        20 CALL IO(IFIN,2,NPTS,COST)
006166      000        30 IF(PRINTM)CALL SHOW(NPTS,COST)
006167      000           CALL TIMOUT
006168      000           RETURN
006169      000           END
006170      000   @ELT,SI TUMER-S*FATHOM.INRAND,,,1341[illegible]052112
006171      000           SUBROUTINE INRAND(NPTS,IFIN,COST,PRINTM)
006172      000   C*          INRAND GENERATES UPPER TRIANGLE OF COST MATRIX RANDOMLY
006173      000   C*          A=LOWER LIMIT , B=UPPER LIMIT
006174      000           PARAMETER MAXPTS=80
006175      000           DIMENSION COST(MAXPTS,MAXPTS),X(*200)
006176      000           LOGICAL PRINTM
006177      000           IF(IFIN.NE.0) GO TO 20
006178      000   C*          GENERATE COST MATRIX
006179      000           READ(5,1000) A,B,X(1)
006180      000    1000   FORMAT( )
006181      000           N=NPTS-1
```

```
006184      000          NUM=(NPTS*NPTS/2)-NPTS/2
006185      000          CALL RANDU(X,NUM)
006184      000          IJ=1
006185      000          DO 10 I=1,N
006186      000          COST(I,I)=1.0E38
006187      000          L=I+1
006188      000          DO 10 J=L,NPTS
006189      000          IJ=IJ+1
006190      000          IYFL=A+(B-A)*X(IJ)
006191      000          COST(I,J)=IYFL
006192      000    10    COST(J,I)=IYFL
006193      000          COST(NPTS,NPTS)=1.0E38
006194      000          GO TO 30
006195      000    20    CALL IO(IFIN,2,NPTS,COST)
006196      000    30    IF(PRINTM) CALL SHOW(NPTS,COST)
006197      000          CALL TIMOUT
006198      000          RETURN
006199      000          END
006200      000    @ELT,SI TUMER-S*FATHOM.10
006201      000          SUBROUTINE IO(IFILE,
006202      000    C*      IO READS OR WRITES A
006203      000    C*      IOPT=1      WRITE
006204      000    C*      IOPT=2      READ
006205      000          PARAMETER MAXPTS=60
006206      000           DIMENSION A(MAXPTS,MAXPTS)
006207      000          LOGICAL BAD
006208      000          BAD=.FALSE.
006209      000        5 CALL NTRAN(IFILE,10,22)
006210      000          DO 10 I=1,N
006211      000          CALL NTRAN(IFILE,IOPT,N,A(1,I),ISTAT,22)
006212      000          IF(ISTAT.LE.0)GO TO 700
006213      000       10 CONTINUE
006214      000          RETURN
006215      000      700 IF(BAD)GO TO 800
006216      000          BAD=.TRUE.
006217      000          GO TO 5
006218      000      800 WRITE(6,2000)IFILE
006219      000     2000 FORMAT(' FILE ERROR ON UNIT',I3)
006220      000          STOP
006221      000          END
006222      000    @ELT,SI TUMER-S*FATHOM.MAIN
006223      000    C*      THIS PROGRAM SOLVES
006224      000    C*      BY FATHOMING
006225      000          INCLUDE BLANK
006226      000          DIMENSION PISAVE(20),IPOUT(20)
006227      000          DIMENSION IPOINT(MAXPTS)
006228      000          DIMENSION ACOST(MAXPTS),TCOST(MAXPTS),ITEMP(MAXPTS)
006229      000          DIMENSION ASAVE(MAXPTS),
006230      000          LOGICAL EXSAVE,PRINTL,
006231      000    C*      READ DATA
006232      000          READ(5,1000)NPTS,ISTART,
006233      000      1000 FORMAT()
006234      000          IF(NPI.EQ.0)GO TO 2
006235      000          READ(5,1000) (IPOUT(I),I=1,NPI)
006236      000          READ(5,1000) (PISAVE(I),I=1,NPI)
006237      000    C*      NPTS=NUMBER OF POINTS
006238      000    C*      ISTART=STARTING POINT (SEARCH IF ISTART=0)
```

```
000239      000   C*      CSAVE=INITIAL CO[illegible] BOUND
000240      000   C*      EPS=STOPPING CR[illegible] DIFFERENCE BETWEEN BOUNDS
000241      000   C*      MAXIT=MAXIMUM NU[illegible] ITERATIONS
000242      000   C*      NPI=NUMBER OF PI [illegible]
000243      000   C*      IFIN=INPUT FILE [illegible] MATRIX
000244      000   C*      IFOUT=OUTPUT FILE [illegible] MATRIX
000245      000   C*      PISAVE=VALUES [illegible] COST MATRIX
000246      000   C*      PRINTL=PRINT OPTION FOR LIST AT EACH STEP
000247      000   C*      PRINTM=PRINT OPTION FOR COST MATRIX
000248      000   C*      PRINTV=PRINT OPTION FOR [illegible] METHOD)
000249      000   C*      MATGEN=COST MATRIX IS [illegible]ERATED IN RANDOM MANNER(=1)
000250      000       2 PRINTL=(L.GT.0)
000251      000         PRINTM=(M.GT.0)
000252      000         PRINTV=(K.GT.0)
000253      000         IF(MATGEN.EQ.1) GO TO 3
000254      000         CALL INPUT(NPTS,[illegible]INTM)
000255      000         GO TO 4
000256      000   3     CALL INRAND(NPTS,[illegible]INTM)
000257      000   C*      FIND STARTING [illegible] COST M[illegible]
000258      000   4     CALL START(ISTART,[illegible]ISAVE,IFOUT,[illegible],PRINTV)
000259      000   C*      READ COST FACTOR [illegible]MING
000260      000         IF(FOUND) GO TO 6[illegible]
000261      000         READ(5,1000,END=90)CPCT
000262      000         CFACT=CPCT/100.0
000263      000         IF(CPCT.LE.0.0)CMAX=[illegible]
000264      000         IF(CPCT.GT.0.0)CMAX=[illegible] CSAVE
000265      000         CMIN=-1.0E30
000266      000         FOUND=.FALSE.
000267      000   C*        PLACE STARTING POINT ON LIST
000268      000       5 NPTSIN=1
000269      000         IPOINT(1)=ISTART
000270      000         LAST=ISTART
000271      000   C*      FIND MINIMUM CO[illegible]TION
000272      000         J1=MINPT(ISTART,1)
000273      000         J2=MINPT2(ISTART,J1)
000274      000   C*      FIND MINIMUM TR[illegible]
000275      000         TCOST(1)=TREE(1,[illegible])
000276      000   C*        COMPUTE COST
000277      000         C=TCOST(1)+COST(J1,[illegible])+COST(J2,ISTART)
000278      000         IF(CMIN.LT.C)CMIN=C
000279      000         IF(PRINTL)CALL OUTPUT([illegible],NT)
000280      000   C*      ADD NEW POINT
000281      000      10 NPTSIN=NPTSIN+1
000282      000         IF(NPTSIN.EQ.NPTS)GO TO 50
000283      000         DO 20 I=1,NPTS
000284      000      20 EX(NPTSIN,I)=.FALSE.
000285      000         IF(NPTSIN.EQ.2)J2=J1
000286      000         ACOST(NPTSIN)=ACOST(NPTSIN-1)+COST(LAST,J1)
000287      000         LAST=J2
000288      000         IPOINT(NPTSIN)=LAST
000289      000         IN(LAST)=.TRUE.
000290      000   C*      FIND MINIMUM CONNECTIONS
000291      000         IF(LAST.NE.J1)GO TO 30
000292      000      25 J1=MINPT(ISTART,1)
000293      000         IF(J1.EQ.0)GO TO 40
000294      000         IF(NPTSIN.NE.1)GO TO 30
000295      000         J2=MINPT2(ISTART,J1)
```

```
006296      000          GO TO 35
006297      000       30 J2=MINPT(LAST,NPTSIN)
006298      000       35 IF(J2.EQ.0)GO TO 40
006299      000    C*       FIND MINIMUM TREE
006300      000          TCOST(NPTSIN)=TREE(NPTSIN,ITEMP)
006301      000    C*       COMPUTE COST
006302      000          C=ACOST(NPTSIN)+TCOST(NPTSIN)+COST(J1,ISTART)+COST(LAST,J2)
006303      000          IF(PRINTL)CALL OUTPUT(NPTS,C,IPOINT)
006304      000          IF(C.LE.CMAX)GO TO 10
006305      000    C*       REMOVE POINT FROM LIST
006306      000       40 NPTSIN=NPTSIN-1
006307      000          IF(NPTSIN.EQ.0)GO TO 80
006308      000          EX(NPTSIN,LAST)=.TRUE.
006309      000          IN(LAST)=.FALSE.
006310      000          LAST=IPOINT(NPTSIN)
006311      000          GO TO 25
006312      000    C*       COMPUTE COST OF COMPLETE LOOP
006313      000       50 C=ACOST(NPTS-1)+COST(J1,[illegible])+COST(LAST,J2)
006314      000          LAST=J2
006315      000          IPOINT(NPTS)=LAST
006316      000          IF(CMAX.LT.C)GO TO 40
006317      000          WRITE(6,2000)
006318      000     2000 FORMAT(//,' NEW SOLUTION')
006319      000          CALL OUTPUT(NPTS,C,IPOINT)
006320      000          CALL TIMOUT
006321      000          IF(CSAVE.LE.C.AND.FOUND)GO TO 40
006322      000          CSAVE=C
006323      000          IF(CPCT.GT.0.0)GO TO 52
006324      000    C*       ADJUST UPPER BOUND
006325      000          CMAX=(C+CMIN)/2.0
006326      000          IF((CMAX-CMIN).LE.EPS)GO TO 90
006327      000          GO TO 55
006328      000       52 CMAX=CFACT*CSAVE
006329      000       55 FOUND=.TRUE.
006330      000          DO 60 I=1,NPTS
006331      000          ISAVE(I)=IPOINT(I)
006332      000          ASAVE(I)=ACOST(I)
006333      000          TSAVE(I)=TCOST(I)
006334      000          DO 60 J=1,NPTS
006335      000       60 EXSAVE(I,J)=EX(I,J)
006336      000          IF(CMAX.LT.CMIN) GO TO 80
006337      000          GO TO 40
006338      000    C*       ADJUST LOWER BOUND
006339      000       80 IF(CPCT.GT.0.0)GO TO 100
006340      000          CMIN=CMAX
006341      000          CALL TIMOUT
006342      000          CMAX=(CMIN+CSAVE)/2.0
006343      000          IF((CMAX-CMIN).LE.EPS)GO TO 90
006344      000          GO TO 105
006345      000       90 GO TO 101
006346      000    C*       PRINT FINAL SOLUTION
006347      000      100 CONTINUE
006348      000          WRITE(6,2100)CPCT
006349      000     2100 FORMAT(//,1X,F5.1,' % SOLUTION')
006350      000          CALL TIMOUT
006351      000          IF(.NOT.ALR) GO TO 101
006352      000          FOUND=.FALSE.
```

```
006353      000          ALR=.FALSE.
006354      000          CSAVE=1.0E38
006355      000          CPCT=0
006356      000    C*      READ COST FACTOR FOR FATHOMING
006357      000    101   READ(5,1000,END=900)CPCT1
006358      000          IF(CPCT1.LE.CPCT) GO TO 101
006359      000           CPCT=CPCT1
006360      000    102   CFACT=CPCT/100.0
006361      000          CMAX=CFACT*CSAVE
006362      000          IF(CMAX.LT.CMIN) GO TO 50
006363      000      105 IF(.NOT.FOUND)GO TO 120
006364      000          DO 110 I=1,NPTS
006365      000          IN(I)=.TRUE.
006366      000          IPOINT(I)=ISAVE(I)
006367      000          ACOST(I)=ASAVE(I)
006368      000          TCOST(I)=TSAVE(I)
006369      000          DO 110 J=1,NPTS
006370      000      110 EX(I,J)=EXSAVE(I,J)
006371      000          LAST=IPOINT(NPTS)
006372      000          NPTSIN=NPTS
006373      000          GO TO 40
006374      000      120 CONTINUE
006375      000          DO 130 I=1,NPTS
006376      000          DO 130 J=1,NPTS
006377      000      130 EX(I,J)=.FALSE.
006378      000          GO TO 5
006379      000    800   WRITE(6,3000)
006380      000    3000  FORMAT(//,' SOLVED ON A ',
006381      000          CPCT=100.0
006382      000          CMIN=CMAX
006383      000          FOUND=.FALSE.
006384      000          GO TO 102
006385      000      900 CALL TIMOUT
006386      000          STOP
006387      000          END
006388      000    @ELT,SI TUMER-S*FATHOM.MINPT,...
006389      000          FUNCTION MINPT(I,N)
006390      000    C*      MINPT FINDS POINT WITH MINIMUM COST FOR CONNECTION TO
006391      000    C*      GIVEN POINT
006392      000          INCLUDE BLANK
006393      000          M=0
006394      000          C=1.0E38
006395      000          DO 10 J=1,NPTS
006396      000          IF(IN(J).OR.EX(N,J))GO TO 10
006397      000          IF(C.LE.COST(I,J))GO TO 10
006398      000          C=COST(I,J)
006399      000          M=J
006400      000       10 CONTINUE
006401      000          MINPT=M
006402      000          RETURN
006403      000          END
006404      000    @ELT,SI TUMER-S*FATHOM.MINPT2,...
006405      000          FUNCTION MINPT2(I,N)
006406      000    C*      MINPT2 FINDS POINT WITH SECOND LOWEST COST FOR CONNECTION TO
006407      000    C*      GIVEN POINT
006408      000    C*       I=GIVEN POINT
006409      000    C*       N=POINT WITH LOWEST COST
```

```
006410  000        INCLUDE BLANK
006411  000        M=0
006412  000        C=1.0E38
006413  000        DO 10 J=1,NPTS
006414  000        IF(J.EQ.I.OR.J.EQ.N)GO TO 10
006415  000        IF(EX(I,J))GO TO 10
006416  000        IF(C.LE.COST(I,J))GO TO 10
006417  000        C=COST(I,J)
006418  000        M=J
006419  000     10 CONTINUE
006420  000        MINPT2=M
006421  000        RETURN
006422  000        END
006423  000  @ELT,SI TUMER-S*FATHOM.ORDER/BATCH,,,200311121411
006424  000        SUBROUTINE ORDER(ISTART)
006425  000  C*      DUMMY SUBROUTINE
006426  000        RETURN
006427  000        END
006428  000  @ELT,SI TUMER-S*FATHOM.ORDER/BATCH,,,170251121411
006429  000        SUBROUTINE ORDER(ISTART)
006430  000  C*      ORDER ORDERS LIST OF COMBINATIONS BY INCREASING COST
006431  000  C*      COMBINATIONS CONTAINING STARTING POINT ARE OMITTED
006432  000        INCLUDE BLANK
006433  000        N=NPTS-1
006434  000        NLIST=(N*N-N)/2
006435  000        IF(ISTART.EQ.0)NLIST=(N*N-N)/2
006436  000        DO 30 K=1,NLIST
006437  000        C=1.0E38
006438  000        DO 20 I=1,N
006439  000        IF(I.EQ.ISTART)GO TO 20
006440  000        L=I+1
006441  000        DO 10 J=L,NPTS
006442  000        IF(J.EQ.ISTART.OR.EX(I,J))GO TO 10
006443  000        IF(C.LE.COST(I,J))GO TO 10
006444  000        C=COST(I,J)
006445  000        LIST(1,K)=I
006446  000        LIST(2,K)=J
006447  000     10 CONTINUE
006448  000     20 CONTINUE
006449  000        I=LIST(1,K)
006450  000        J=LIST(2,K)
006451  000     30 EX(I,J)=.TRUE.
006452  000        DO 40 I=1,NPTS
006453  000        DO 40 J=1,NPTS
006454  000     40 EX(I,J)=.FALSE.
006455  000        RETURN
006456  000        END
006457  000  @ELT,SI TUMER-S*FATHOM.OUTPUT/BATCH,,,172454113311
006458  000        SUBROUTINE OUTPUT(N,C,IPOINT)
006459  000  C*       OUTPUT PRINTS COST AND LIST OF POINTS
006460  000        DIMENSION IPOINT(1)
006461  000        M=MIN(N,25)
006462  000        WRITE(6,2000)C,(IPOINT(I),I=1,M)
006463  000        IF(N.LE.25)GO TO 900
006464  000        DO 10 L=26,N,25
006465  000        M=MIN(N,L+24)
006466  000     10 WRITE(6,2100) (IPOINT(I),I=L,M)
```

```
 2000 FORMAT(' COST',F7.0,[illegible],25I4)
 2100 FORMAT(22X,25I4)
  900 RETURN
      END
@ELT,SI TUMER-S*FATHOM.[illegible],17261411[illegible]
      SUBROUTINE OU[illegible]
C*       OUTPUT PRINTS [illegible] OF POINTS
      DIMENSION IPOINT(1)
      M=MIN(N,12)
      WRITE(6,2000)C,(IPOIN[illegible],I=1,M)
      IF(N.LE.12)GO TO 900
      DO 10 L=13,N,12
      M=MIN(N,L+11)
   10 WRITE(6,2100) (IPOI[illegible],I=L,M)
 2000 FORMAT(' COST',F7.0,[illegible],12I4)
 2100 FORMAT(22X,12I4)
  900 RETURN
      END
@ELT,SI TUMER-S*FATHOM.[illegible]13470605101[illegible]
      SUBROUTINE SHOW([illegible])
C*       SHOW PRINTS COST [illegible]
      PARAMETER MAXPTS=[illegible]
      DIMENSION COST(MAXPTS,MA[illegible])
      WRITE(6,2000)
 2000 FORMAT(' COST MATRIX')
      DO 20 K=1,NPTS,10
      L=MIN(NPTS,K+15)
      WRITE(6,2100) (J,J=K,L)
      DO 20 I=1,NPTS
   20 WRITE(6,2200)I,(COST(I,J),J=K,L)
 2100 FORMAT(4X,16I7)
 2200 FORMAT(1X,I3,16F7.0)
      RETURN
      END
@ELT,SI TUMER-S*FATHOM.SHOW[illegible]13503605101[illegible]
      SUBROUTINE SHOW(N[illegible])
C*       SHOW PRINTS COST MATRIX
      PARAMETER MAXPTS=60
      DIMENSION COST(MAXPTS,MAXPTS)
      WRITE(6,2000)
 2000 FORMAT(' COST MATRIX')
      DO 20 K=1,NPTS,9
      L=MIN(NPTS,K+8)
      WRITE(6,2100) (J,J=K,L)
      DO 20 I=1,NPTS
   20 WRITE(6,2200)I,(COST(I,J),J=K,L)
 2100 FORMAT(4X,9I7)
 2200 FORMAT(1X,I3,9F7.0)
      RETURN
      END
@ELT,SI TUMER-S*FATHOM.START,,1[illegible]23070312
      SUBROUTINE START(ISTART,MAXIT,NPT,PISAVE,IPOUT,PRINTM,PRINTV)
C*       START FINDS THE STARTING POINT AND ADJUSTS THE COST MATRIX
      INCLUDE BLANK
      DIMENSION PISAVE(1),IPOUT(1)
      DIMENSION IPI(MAXPTS),ITEMP(MAXPTS),CTEMP(MAXPTS,MAXPTS)
      LOGICAL PRINTM,PRINTV
```

```
006524      000    C*      ORDER CONNECTIONS BY INCREASING COST
006525      000           CALL ORDER(0)
006526      000    C*      FIND STARTING POINT WITH MAXIMUM LOWER BOUND OF COST
006527      000           CMAX=-1.0E38
006528      000           JSTART=ISTART
006529      000           DO 20 I=1,NPTS
006530      000           IF(JSTART.NE.0.AND.JSTART.NE.I)GO TO 20
006531      000           IN(I)=.TRUE.
006532      000           T=TREE(1,ITEMP)
006533      000           K1=MINPT(I,1)
006534      000           K2=MINPT2(I,K1)
006535      000           C=T+COST(K1,I)+COST(I,K2)
006536      000           IF(C.LE.CMAX)GO TO 20
006537      000           ISTART=I
006538      000           CMAX=C
006539      000           J1=K1
006540      000           J2=K2
006541      000           DO 10 J=1,NPTS
006542      000        10 IPI(J)=ITEMP(J)
006543      000        20 IN(I)=.FALSE.
006544      000           WRITE(6,2000) CMAX,ISTART
006545      000      2000 FORMAT(' COST',F10.0,' STARTING POINT',I3)
006546      000           IN(ISTART)=.TRUE.
006547      000           ISAVE(1)=ISTART
006548      000    C*       USE VOGEL METHOD TO FIND SOLUTION
006549      000           IF(CSAVE.LE.0.0)CALL VOGEL(NPTS,COST,PRINTV,CSAVE,ISAVE)
006550      000           IF(NPI.LE.0)GO TO 900
006551      000           ICMIN=NPTS**2
006552      000           CALL COPY(NPTS,COST,CTEMP)
006553      000           CALL TIMOUT
006554      000    C*      ITERATE ON COST MATRIX TO FIND MAXIMUM LOWER BOUND
006555      000           JPI=1
006556      000           PI=PISAVE(1)
006557      000           WRITE(6,2100)PI
006558      000     2100 FORMAT(' PI=',F10.4)
006559      000           N=NPTS-1
006560      000           IT=0
006561      000    C*      ADJUST COST
006562      000        25 IPI(ISTART)=0
006563      000           IPI(J1)=IPI(J1)+1
006564      000           IPI(J2)=IPI(J2)+1
006565      000           ICOST=ABS(IPI(NPTS))
006566      000           ICSQ=ICOST**2
006567      000           DO 30 J=1,N
006568      000           ICOST=ICOST+ABS(IPI(J))
006569      000           ICSQ=ICSQ+IPI(J)**2
006570      000           L=J+1
006571      000           DO 30 K=L,NPTS
006572      000           COST(J,K)=COST(J,K)+PI*(IPI(J)+IPI(K))
006573      000        30 COST(K,J)=COST(J,K)
006574      000           IF(ICOST.EQ.0)GO TO 800
006575      000           IF(PRINTM)CALL SHOW(NPTS,COST)
006576      000    C*      REORDER CONNECTIONS
006577      000           CALL ORDER(ISTART)
006578      000    C*      COMPUTE MINIMUM COST
006579      000           T=TREE(1,IPI)
006580      000           J1=MINPT(ISTART,1)
```

```
000581      000          J2=MINPT2(ISTART,J1)
000582      000          C=T+COST(J1,ISTART)+COST(ISTART,J2)
000583      000          IF(C.LE.CMAX)GO TO 40
000584      000    C*      NEW MAXIMUM FOUND
000585      000          IT=0
000586      000          CMAX=C
000587      000          CALL COPY(NPTS,COST,CTEMP)
000588      000          K1=J1
000589      000          K2=J2
000590      000          DO 35 J=1,NPTS
000591      000       35 ITEMP(J)=IPI(J)
000592      000          GO TO 25
000593      000       40 IT=IT+1
000594      000          IF(IT.LT.MAXIT)GO TO 25
000595      000    C*      GO BACK TO BEST COST
000596      000          CALL COPY(NPTS,CTEMP,COST)
000597      000          IF(IFOUT(JPI).NE.0)CALL SAVOUT(JPI),I,NPTS,COST)
000598      000          WRITE(6,2001) CMAX
000599      000          CALL TIMOUT
000600      000          IF(JPI.EQ.NPI)GO TO 800
000601      000          IT=0
000602      000          JPI=JPI+1
000603      000          PI=PISAVE(JPI)
000604      000          WRITE(6,2100)PI
000605      000          J1=K1
000606      000          J2=K2
000607      000          DO 50 J=1,NPTS
000608      000       50 IPI(J)=ITEMP(J)
000609      000          GO TO 25
000610      000   800    FOUND=.TRUE.
000611      000   900    WRITE(6,2000) CSAVE,FOUND
000612      000          WRITE(6,2001) CMAX
000613      000   2001   FORMAT(' MAXIMUM COST',F10.0)
000614      000          CALL TIMOUT
000615      000          RETURN
000616      000          END
000617      000   @ELT,SI TUMER-S*FATHOM,TIMOUT,,,1757610371
000618      000          SUBROUTINE TIMOUT
000619      000          K=ITIME(I,J)
000620      000          T=I/5000.
000621      000          WRITE(6,2000)T
000622      000    2000  FORMAT(' TIME',F10.3,' SEC')
000623      000          RETURN
000624      000          END
000625      000   @ELT,SI TUMER-S*FATHOM,TREE,,,
```

```
000636      000          IF(IN(I).OR.I.EQ.NPTS)GO TO 90
000639      000          L=I+1
000640      000          DO 10 J=L,NPTS
000641      000          IF(IN(J))GO TO 10
000642      000          IF(C.LE.COST(I,J))GO TO 20
000643      000          IROW=I
000644      000          ICOL=J
000645      000          C=COST(I,J)
000646      000       10 CONTINUE
000647      000       20 LINK(I)=0
000648      000          IPI(IROW)=1+IPI(IROW)
000649      000          IPI(ICOL)=1+IPI(ICOL)
000650      000          M=M-1
000651      000          IF(M.EQ.0)GO TO 900
000652      000    C*      FIND MINIMUM CONNECTION TO EACH POINT
000653      000          CMIN=1.0E38
000654      000          DO 50 I=1,NPTS
000655      000          IF(IN(I))GO TO 50
000656      000          IF(I.EQ.IROW.OR.I.EQ.ICOL)GO TO 50
000657      000          IF(COST(IROW,I).GT.COST(ICOL,I))GO TO 30
000658      000          LINK(I)=IROW
000659      000          CLINK(I)=COST(IROW,I)
000660      000          GO TO 40
000661      000       30 LINK(I)=ICOL
000662      000          CLINK(I)=COST(ICOL,I)
000663      000       40 IF(CLINK(I).GE.CMIN)GO TO 50
000664      000          JROW=LINK(I)
000665      000          JCOL=I
000666      000          CMIN=CLINK(I)
000667      000       50 CONTINUE
000668      000          IPI(JROW)=1+IPI(JROW)
000669      000          IPI(JCOL)=1+IPI(JCOL)
000670      000          ICOL=JCOL
000671      000          LINK(ICOL)=0
000672      000          C=C+CLINK(ICOL)
000673      000          M=M-1
000674      000          IF(M.EQ.0)GO TO 900
000675      000    C*      UPDATE MINIMUM CONNECTION
000676      000          DO 80 J=1,M
000677      000          CMIN=1.0E38
000678      000          DO 70 I=1,NPTS
000679      000          IF(LINK(I).EQ.0)GO TO 70
000680      000          IF(CLINK(I).LE.COST(I,ICOL))GO TO 60
000681      000          LINK(I)=ICOL
000682      000          CLINK(I)=COST(I,ICOL)
000683      000       60 IF(CLINK(I).GE.CMIN)GO TO 70
000684      000          JROW=LINK(I)
000685      000          JCOL=I
000686      000          CMIN=CLINK(I)
000687      000       70 CONTINUE
000688      000          IPI(JROW)=1+IPI(JROW)
000689      000          IPI(JCOL)=1+IPI(JCOL)
000690      000          ICOL=JCOL
000691      000          LINK(ICOL)=0
000692      000       80 C=C+CLINK(ICOL)
000693      000      900 TREE=C
000694      000          RETURN
```

```
      END
@ELT,SI TUMER-S*FATHOM,IREF/OLD,,,1/0314121411
      FUNCTION TREE(N,IPI)
C*      TREE FINDS THE COST OF A MINIMUM TREE
      INCLUDE BLANK
      DIMENSION IPI(1)
      DIMENSION LINK(MAXPTS)
      M=NPTS-N-1
      NLINK=0
      C=0.0
      IF(M.LE.0)GO TO 900
      DO 10 I=1,NPTS
      IPI(I)=-2
   10 LINK(I)=0
      DO 60 I=1,NLIST
C*      CHECK IF POINTS IN LIST
      J=LIST(1,I)
      K=LIST(2,I)
      IF(IN(J).OR.IN(K))GO TO 40
C*      CHECK IF POINTS ARE JUST CONNECTED
      MX=MAX(LINK(J),LINK(K))
      IF(MX.EQ.0)GO TO 40
      IF(LINK(J).EQ.LINK(K))GO TO 60
C*      ADD BRANCH TO TREE
      MN=MIN(LINK(J),LINK(K))
      IF(MN.EQ.0)GO TO 30
C*      COMBINE LINKS
      DO 20 L=1,NPTS
      IF(LINK(L).EQ.MX)LINK(L)=MN
   20 CONTINUE
      GO TO 50
C*      ADD TO EXISTING LINK
   30 LINK(J)=MX
      LINK(K)=MX
      GO TO 50
C*      FORM NEW LINK
   40 NLINK=NLINK+1
      LINK(J)=NLINK
      LINK(K)=NLINK
   50 C=C+COST(J,K)
      IPI(J)=1+IPI(J)
      IPI(K)=1+IPI(K)
      M=M-1
      IF(M.EQ.0)GO TO 900
   60 CONTINUE
  900 TREE=C
      RETURN
      END
@ELT,SI TUMER-S*FATHOM,VOGEL,,,1350213051012
      SUBROUTINE VOGELIN(COST,MAXPTS,NPTS,V,Z,ISAVE)
      PARAMETER MAXPTS=50
      DIMENSION COST(MAXPTS,MAXPTS),ISAVE(MAXPTS)
      DIMENSION LROW(MAXPTS),LCOL(MAXPTS),ITOUP(MAXPTS),NODE(MAXPTS)
      LOGICAL PRINTV
      CMAX=1.0E38
C*** INITIALIZE VARIABLES
      ICOUNT=1
```

```
000752      000          ITREE=0
000753      000          DO 15 I=1,N
000754      000          LROW(I)=0
000755      000          LCOL(I)=0
000756      000          NODE(I)=0
000757      000       15 CONTINUE
000758      000          Z=0.
000759      000       16 CONTINUE
000760      000          ROWPEN=0.
000761      000          COLPEN=0.
000762      000    C*** FIGURE ROW PENALTIES
000763      000          DO 50 I=1,N
000764      000          IF (LROW(I).EQ.1) GO TO 50
000765      000          CMIN=CMAX
000766      000          DO 20 J=1,N
000767      000          IF(I.EQ.J)GO TO 20
000768      000          IF(ICOUNT.NE.N.AND.NODE(I).EQ.NODE(J).AND.NODE(I).NE.0) GO TO 20
000769      000          IF(LCOL(J).EQ.1.OR.COST(I,J).GT.CMIN) GO TO 20
000770      000          MINI=J
000771      000          CMIN=COST(I,J)
000772      000       20 CONTINUE
000773      000          DIFROW=CMAX
000774      000          DIFF=CMAX
000775      000          DO 25 J=1,N
000776      000          IF(I.EQ.J)GO TO 25
000777      000          IF(ICOUNT.NE.N.AND.NODE(I).EQ.NODE(J).AND.NODE(I).NE.0) GO TO 25
000778      000          IF(LCOL(J).EQ.1.OR.COST(I,J).GT.CMIN) GO TO 25
000779      000          DIFF=IABS(COST(I,J)-CMIN)
000780      000          IF(DIFROW.GT.DIFF) DIFROW=DIFF
000781      000       25 CONTINUE
000782      000          IF(ROWPEN.GE.DIFROW) GO TO 50
000783      000          ROWPEN=DIFROW
000784      000          IROW=I
000785      000          JROW=MINI
000786      000       50 CONTINUE
000787      000    C***  FIGURE COLUMN PENALTIES
000788      000          DO 100 J=1,N
000789      000          IF(LCOL(J).EQ.1) GO TO 100
000790      000          CMIN=CMAX
000791      000          DO 60 I=1,N
000792      000          IF(I.EQ.J)GO TO 60
000793      000          IF(ICOUNT.NE.N.AND.NODE(I).EQ.NODE(J).AND.NODE(J).NE.0) GO TO 60
000794      000          IF(LROW(I).EQ.1.OR.COST(I,J).GT.CMIN) GO TO 60
000795      000          MINI=I
000796      000          CMIN=COST(I,J)
000797      000       60 CONTINUE
000798      000          DIFCOL=CMAX
000799      000          DIFF=CMAX
000800      000          DO 65 I=1,N
000801      000          IF(I.EQ.J)GO TO 65
000802      000          IF(ICOUNT.NE.N.AND.NODE(I).EQ.NODE(J).AND.NODE(J).NE.0) GO TO 65
000803      000          IF(LROW(I).EQ.1.OR.LCOL(I).EQ.1) GO TO 65
000804      000          DIFF=ABS(COST(I,J)-CMIN)
000805      000          IF(DIFCOL.GT.DIFF) DIFCOL=DIFF
000806      000       65 CONTINUE
000807      000          IF(COLPEN.GE.DIFCOL) GO TO 100
000808      000          COLPEN=DIFCOL
```

```
000809      000          ICOL=MINI
000810      000          JCOL=J
000811      000        100 CONTINUE
000812      000      C*** WE NOW HAVE ROW AND COLUMN MULTIPLES.
000813      000      C*** DELETE THE ROW & THE COLUMN CONTAINING THE MAX ELEMENT
000814      000          IF(COLPEN.GT.ROWPEN) GO TO 105
000815      000          Z=Z+COST(IROW,JROW)
000816      000          LROW(IROW)=1
000817      000          LCOL(JROW)=1
000818      000          COST(JROW,IROW)=CMAX
000819      000          ITOUR(IROW)=JROW
000820      000          IF(PRINTV)WRITE(6,1020) ICOUNT,IROW,JROW
000821      000          IF(NODE(IROW).NE.0.OR.NODE(JROW).NE.0) GO TO 106
000822      000          ITREE=ITREE+1
000823      000          NODE(IROW)=ITREE
000824      000          NODE(JROW)=ITREE
000825      000          GO TO 115
000826      000        105 CONTINUE
000827      000          IF(NODE(IROW).EQ.0.OR.NODE(JROW).EQ.0) GO TO 110
000828      000          ITOP=MAX(NODE(IROW),NODE(JROW))
000829      000          IREP=MIN(NODE(IROW),NODE(JROW))
000830      000          GO TO 113
000831      000        106 CONTINUE
000832      000          IF(NODE(IROW).EQ.0) NODE(IROW)=NODE(JROW)
000833      000          NODE(JROW)=NODE(IROW)
000834      000          GO TO 115
000835      000        110 Z=Z+COST(ICOL,JCOL)
000836      000          LROW(ICOL)=1
000837      000          LCOL(JCOL)=1
000838      000          COST(JCOL,ICOL)=CMAX
000839      000          ITOUR(ICOL)=JCOL
000840      000          IF(PRINTV)WRITE(6,1020) ICOUNT,ICOL,JCOL
000841      000          IF(NODE(ICOL).NE.0.OR.NODE(JCOL).NE.0) GO TO 111
000842      000          ITREE=ITREE+1
000843      000          NODE(ICOL)=ITREE
000844      000          NODE(JCOL)=ITREE
000845      000          GO TO 115
000846      000        111 CONTINUE
000847      000          IF(NODE(ICOL).EQ.0.OR.NODE(JCOL).EQ.0) GO TO 112
000848      000          ITOP=MAX(NODE(ICOL),NODE(JCOL))
000849      000          IREP=MIN(NODE(ICOL),NODE(JCOL))
000850      000          GO TO 113
000851      000        112 CONTINUE
000852      000          IF(NODE(ICOL).EQ.0) NODE(ICOL)=NODE(JCOL)
000853      000           NODE(JCOL)=NODE(ICOL)
000854      000          GO TO 115
000855      000        113 DO 114 I=1,N
000856      000            IF(NODE(I).EQ.ITOP) NODE(I)=IREP
000857      000        114 CONTINUE
000858      000        115 IF(ICOUNT.EQ.N) GO TO 999
000859      000       1020 FORMAT(1X,I3,1H.,' CITY ',I3,' TO CITY ',I3)
000860      000          ICOUNT=ICOUNT+1
000861      000      C     DO 130 I=1,N
000862      000      C     IF(PRINTV)WRITE(6,1015) (COST(I,J),J=1,N)
000863      000      C 130 CONTINUE
000864      000          GO TO 16
000865      000        999 CONTINUE
```

```
000866  000        I=ISAVE(1)
000867  000        DO 200 J=2,N
000868  000        I=ITOUR(I)
000869  000      200 ISAVE(J)=I
000870  000        CALL OUTPUT(N,Z,ISAVE)
000871  000    C*** RESTORE MATRIX
000872  000        DO 300 I=1,N
000873  000        J=ITOUR(I)
000874  000      300 COST(J,I)=COST(I,J)
000875  000        RETURN
000876  000        END
000877  000    @PDP,FI TUMER-S*FATHOM.COMMON/BLANK,,,150624070312
000878  000    BLANK   PROC
000879  000        PARAMETER MAXPTS=60,MXLIST=3160
000880  000        COMMON NPTS,NLIST,CSAVE,ITOUR(MAXPTS),LIST(2,MXLIST),
000881  000       *   ISAVE(MAXPTS),COST(MAXPTS,MAXPTS),EX(MAXPTS,MAXPTS)
000882  000       *,FOUND
000883  000        LOGICAL IN,EX,FOUND
000884  000     END
000885  000    @ELT,SI TUMER-S*NONSYM.CITY,,,140833070512
000886  000    @ASG,A TUMER-S*NONSYM.
000887  000    @ASG,A TUMER-S*MATRIX.
000888  000    @USE 9,MATRIX.
000889  000    @ADD NONSYM.MAP-XQT
000890  000    9,0,1.0E30,0.,20,3,0,0,0,0,0
000891  000    0,0,9
000892  000    5.,1.,0.2
000893  000    @ADD FATHOM-DATA.CITY/9
000894  000    92.
000895  000    95.
000896  000    97.
000897  000    100.
000898  000    @EOF
000899  000    @ELT,SI TUMER-S*NONSYM.CMINP3,,,120154063012
000900  000        FUNCTION CMINP3(N,J1,J2)
000901  000    C*          FIND COST OF BEST CONNECTION
000902  000    C*          FROM N TO THE ONE TOUR
000903  000    C*          J1 FROM N , J2 TO N
000904  000        INCLUDE BLANK
000905  000        J2=MINPT(N,1)
000906  000        IF(J2.NE.0) GO TO 5
000907  000        C01=1.0E38
000908  000        GO TO 40
000909  000    5   C=1.0E38
000910  000        DO 10 J=1,NPTS
000911  000        IF(J.EQ.N.OR.J.EQ.J2) GO TO 10
000912  000        IF(C.LE.COST(J,N)) GO TO 10
000913  000        C=COST(J,N)
000914  000        J1=J
000915  000    10  CONTINUE
000916  000        C01=COST(N,J2)+COST(J1,N)
000917  000        C=1.0E38
000918  000        DO 20 J=1,NPTS
000919  000        IF(J.EQ.N) GO TO 20
000920  000        IF(C.LE.COST(J,N)) GO TO 20
000921  000        C=COST(J,N)
000922  000        J4=J
```

```
006925      000    20    CONTINUE
006924      000          J3=0
006925      000          C=1.0E38
006926      000          DO 30 J=1,NPTS
006927      000          IF(J.EQ.N.OR.J.EQ.[illegible]) GO TO 30
006928      000          IF(C.LE.COST(N,J)) GO TO 30
006929      000          C=COST(N,J)
006930      000          J3=J
006931      000    30    CONTINUE
006932      000          IF(J3.EQ.0) GO TO 40
006933      000          C02=COST(N,J3)+COST(J3,1)
006934      000          IF(C01.LE.C02) GO TO 40
006935      000          C01=C02
006936      000          J1=J4
006937      000          J2=J3
006938      000    40    CMINP3=C01
006939      000          RETURN
006940      000          END
006941      000    @ELT,SI TUMER-S*NONSYM.COPY[illegible]
006942      000          SUBROUTINE COPY(N,A,B)
006943      000    C*       COPY COPIES ONE ARRAY INTO ANOTHER
006944      000          PARAMETER MAXPTS=200
006945      000          DIMENSION A(MAXPTS,MAXPTS),B(MAXPTS,MAXPTS)
006946      000          DO 10 I=1,N
006947      000          DO 10 J=1,N
006948      000       10 B(I,J)=A(I,J)
006949      000          RETURN
006950      000          END
006951      000    @ELT,SI TUMER-S*NONSYM.FOR[illegible]
006952      000    @FOR,Y NONSYM.TIMOUT
006953      000    @FOR,Y NONSYM.IO
006954      000    @FOR,Y NONSYM.COPY
006955      000    @FOR,Y NONSYM.MINPT
006956      000    @FOR,Y NONSYM.SHOW/BATCH,.SHOW
006957      000    @FOR,Y NONSYM.INPUT
006958      000    @FOR,Y NONSYM.MAIN
006959      000    @FOR,Y NONSYM.OUTPUT/BATCH,.OUTPUT
006960      000    @FOR,Y NONSYM.START
006961      000    @FOR,Y NONSYM.CMINP3
006962      000    @FOR,Y NONSYM.TREE
006963      000    @FOR,Y NONSYM.INRAND
006964      000    @FOR,Y NONSYM.MINPT2
006965      000    @EOF
006966      000    @ELT,SI TUMER-S*NONSYM.INPUT[illegible]
006967      000          SUBROUTINE INPUT(NPTS,[illegible],COST,PRINTM)
006968      000    C*       INPUT READS [illegible] COST MATRIX
006969      000          PARAMETER MAXPTS=[illegible]
006970      000          DIMENSION COST(MAXPTS,MAXPTS)
006971      000          LOGICAL PRINTM
006972      000          IF(IFIN.NE.0)GO TO 20
006973      000    C*       READ MATRIX
006974      000          DO 10 I=1,NPTS
006975      000    10    READ(5,1000) (COST(I,J),J=1,NPTS)
006976      000     1000 FORMAT()
006977      000          GO TO 30
006978      000    C*     USE MATRIX PREVIOUSLY SAVED
006979      000       20 CALL IO(IFIN,2,NPTS,COST)
```

```
000980  000       30 IF(PRINTM)CALL SHOW(NPTS,COST)
000981  000          CALL TIMOUT
000982  000          RETURN
000983  000          END
000984  000    @ELT,SI TUMER-S*NONSYM.IN[illegible]1063212
000985  000          SUBROUTINE INRAND(NPTS,[illegible]COST,PRINTM)
000986  000    C*       INRAND GENERATES [illegible] COST MATRIX RANDOMLY
000987  000    C*       A=LOWER LIMIT , [illegible] LIMIT
000988  000          PARAMETER MAXPTS=60
000989  000          DIMENSION COST(MAXPTS,MAXPTS),X(6325)
000990  000          LOGICAL PRINTM
000991  000          IF(IFIN.NE.0) GO TO 20
000992  000    C*       GENERATE COST MATRIX
000993  000          READ(5,1000) A,B,X(1)
000994  000    1000  FORMAT( )
000995  000          NUM=(NPTS*NPTS)-NPTS+1
000996  000          CALL RANDU(X,NUM)
000997  000          IJ=1
000998  000          DO 10 I=1,NPTS
000999  000          COST(I,I)=1.0E38
001000  000          DO 10 J=1,NPTS
001001  000          IF(I.EQ.J) GO TO 10
001002  000          IJ=IJ+1
001003  000          IYFL=A+(B-A)*X(IJ)
001004  000          COST(I,J)=IYFL
001005  000    10    CONTINUE
001006  000          GO TO 30
001007  000    20    CALL IO(IFIN,2,NPTS,COST)
001008  000    30    IF(PRINTM) CALL SHOW(NPTS,COST)
001009  000          CALL TIMOUT
001010  000          RETURN
001011  000          END
001012  000    @ELT,SI TUMER-S*NONSYM.IO[illegible]1012
001013  000          SUBROUTINE IO(IFILE,IOPT,N,A)
001014  000    C*       IO READS OR WRITES A MATRIX IN A FILE
001015  000    C*       IOPT=1       WRITE
001016  000    C*       IOPT=2       READ
001017  000          PARAMETER MAXPTS=60
001018  000           DIMENSION A(MAXPTS,MAXPTS)
001019  000          LOGICAL BAD
001020  000          BAD=.FALSE.
001021  000        5 CALL NTRAN(IFILE,10,22)
001022  000          DO 10 I=1,N
001023  000          CALL NTRAN(IFILE,IOPT,N,A(1,I),ISTAT,22)
001024  000          IF(ISTAT.LE.0)GO TO 700
001025  000       10 CONTINUE
001026  000          RETURN
001027  000      700 IF(BAD)GO TO 800
001028  000          BAD=.TRUE.
001029  000          GO TO 5
001030  000      800 WRITE(6,2000)IFILE
001031  000     2000 FORMAT(' FILE ERROR ON UNIT',I3)
001032  000           STOP
001033  000          END
001034  000    @ELT,SI TUMER-S*NONSYM.MAIN,,,172150100512
001035  000    C*       THIS PROGRAM SOLVES THE TRAVELING SALESMAN PROBLEM
001036  000    C*       BY FATHOMING
```

```
007037      000          INCLUDE BLANK
007038      000          DIMENSION PISAVE(20),ITOUR(50)
007039      000          DIMENSION IPOINT(MAXPTS),
007040      000          DIMENSION ACOST(MAXPTS,MAXPTS),
007041      000          DIMENSION ASAVE(MAXPTS,MAXPTS),
007042      000          LOGICAL EXSAVE,PRINTL,PRINTM,PRINTV,ALR
007043      000    C*      READ DATA
007044      000          READ(5,1000)NPTS,ISTART,CSAVE,EPS,MAXIT,NPI,IFIN,L,M,K,MATGEN
007045      000          WRITE(6,1001) NPTS,ISTART,CSAVE,EPS,MAXIT,NPI
007046      000    1001  FORMAT(//2X,'NPTS=',I5,'  ISTART=',I4,'  EPS=',F6.2,
007047      000         *'  MAXIT=',I5,'  NPI=',I4)
007048      000    1002  FORMAT(/,2X,' PI VALUES ARE',10F6.1))
007049      000     1000 FORMAT()
007050      000          IF(NPI.EQ.0)GO TO 2
007051      000          READ(5,1000) (IPOINT(I),I=1,NPI)
007052      000          READ(5,1000) (PISAVE(I),I=1,NPI)
007053      000    C*      NPTS=NUMBER OF POINTS
007054      000    C*      ISTART=STARTING POINT FOR SEARCH IF ISTART>0
007055      000    C*      CSAVE=INITIAL COST VALUE (IF KNOWN)
007056      000    C*      EPS=STOPPING CRITERION FOR DIFFERENCE BETWEEN BOUNDS
007057      000    C*      MAXIT=MAXIMUM NUMBER OF ITERATIONS
007058      000    C*      NPI=NUMBER OF PI VALUES
007059      000    C*      IFIN=INPUT FILE FOR COST MATRIX
007060      000    C*      IFOUT=OUTPUT FILE FOR COST MATRIX
007061      000    C*      PISAVE=VALUES FOR MODIFYING COST MATRIX
007062      000    C*      PRINTL=PRINT OPTION FOR COST AT EACH STEP
007063      000    C*      PRINTM=PRINT OPTION FOR COST MATRIX
007064      000    C*      PRINTV=PRINT OPTION FOR ALR METHOD
007065      000    C*      MATGEN=COST MATRIX IS GENERATED IN RANDOM MANNER(=1)
007066      000        2 PRINTL=(L.GT.0)
007067      000          PRINTM=(M.GT.0)
007068      000          PRINTV=(K.GT.0)
007069      000          IF(MATGEN.EQ.1) GO TO 3
007070      000          CALL INPUT(NPTS,IFIN,CSAVE,PRINTM)
007071      000          GO TO 4
007072      000    3     CALL INRAND(NPTS,IFIN,CSAVE,PRINTM)
007073      000    C*      FIND STARTING POINT AND ADJUST COST MATRIX
007074      000    4     FOUND=.FALSE.
007075      000          CALL START(ISTART,NPTS,NPI,PISAVE,IFOUT,PRINTM,PRINTV)
007076      000          IF(FOUND) GO TO 800
007077      000    C*      READ COST FACTOR FOR FATHOMING
007078      000          READ(5,1000,END=800)CPCT
007079      000          CFACT=CPCT/100.0
007080      000          CMIN=CMAX
007081      000          IF(CPCT.LE.0.0)CMAX=CSAVE
007082      000          IF(CPCT.GT.0.0)CMAX=(1.+CFACT)*CSAVE
007083      000          FOUND=.FALSE.
007084      000    C*       PLACE STARTING POINT ON LIST
007085      000        5 NPTSIN=1
007086      000          IPOINT(1)=ISTART
007087      000          LAST=ISTART
007088      000    C*      FIND MINIMUM CONNECTION
007089      000          C=CMINP3(ISTART,J1,J2)
007090      000    C*      FIND MINIMUM TREE
007091      000          TCOST(1)=TREE(1,ITEMP,ITEMPO)
007092      000    C*       COMPUTE COST
007093      000          C=C+TCOST(1)
```

```
000094      000            IF(CMIN.LT.C)CMIN=C
000095      000            IF(PRINTL)CALL OUTPUT(1,C,IPOINT)
000096      000      C*      ADD NEW POINT
000097      000         10 NPTSIN=NPTSIN+1
000098      000            IF(NPTSIN.EQ.NPTS)GO TO 50
000099      000            DO 20 I=1,NPTS
000100      000         20 EX(NPTSIN,I)=.FALSE.
000101      000            ACOST(NPTSIN)=ACOST(NPTSIN-1)+COST(LAST,J2)
000102      000            LAST=J2
000103      000            IPOINT(NPTSIN)=LAST
000104      000            IN(LAST)=.TRUE.
000105      000      C*      FIND MINIMUM CONNECTIONS
000106      000         25 IF(NPTSIN.NE.1) GO TO 30
000107      000            C=CMINP3(ISTART,J1,J2)
000108      000            GO TO 35
000109      000         30 J1=MINPT2(ISTART)
000110      000            J2=MINPT(LAST,NPTSIN)
000111      000         35 IF(J2.EQ.0)GO TO 40
000112      000      C*       FIND MINIMUM TREE
000113      000            TCOST(NPTSIN)=TREL(NPTSIN,J2,TEMP0)
000114      000      C*      COMPUTE COST
000115      000            C=ACOST(NPTSIN)+TCOST(NPTSIN)+COST(J1,ISTART)+COST(LAST,J2)
000116      000            IF(PRINTL)CALL OUTPUT(NPTSIN,C,IPOINT)
000117      000            IF(C.LE.CMAX)GO TO 10
000118      000      C*      REMOVE POINT FROM LIST
000119      000         40 NPTSIN=NPTSIN-1
000120      000            IF(NPTSIN.EQ.0)GO TO 80
000121      000            EX(NPTSIN,LAST)=.TRUE.
000122      000            IN(LAST)=.FALSE.
000123      000            LAST=IPOINT(NPTSIN)
000124      000            GO TO 25
000125      000      C*      COMPUTE COST OF COMPLETE TOUR
000126      000         50 C=ACOST(NPTS-1)+COST(J1,ISTART)+COST(LAST,J2)
000127      000            LAST=J2
000128      000            IPOINT(NPTS)=LAST
000129      000            IF(CMAX.LT.C)GO TO 40
000130      000            WRITE(6,2000)
000131      000       2000 FORMAT(//,' NEW SOLUTION')
000132      000            CALL OUTPUT(NPTS,C,IPOINT)
000133      000            CALL TIMOUT
000134      000            IF(CSAVE.LE.C.AND.FOUND)GO TO 40
000135      000            CSAVE=C
000136      000            IF(CPCT.GT.0.0)GO TO 52
000137      000      C*      ADJUST UPPER BOUND
000138      000            CMAX=(C+CMIN)/2.0
000139      000            IF((CMAX-CMIN).LE.EPS)GO TO 90
000140      000            GO TO 55
000141      000         52 CMAX=CFACT*CSAVE
000142      000         55 FOUND=.TRUE.
000143      000            DO 60 I=1,NPTS
000144      000            ISAVE(I)=IPOINT(I)
000145      000            ASAVE(I)=ACOST(I)
000146      000            TSAVE(I)=TCOST(I)
000147      000            DO 60 J=1,NPTS
000148      000         60 EXSAVE(I,J)=EX(I,J)
000149      000            IF(CMAX.LT.CMIN) GO TO 80
000150      000            GO TO 40
```

```
007151      000   C*       ADJUST LOWER BOUND
007152      000      80 IF(CPCT.GT.0.0)GO TO 100
007153      000         CMIN=CMAX
007154      000         CMAX=(CMIN+CSAVE)/2.0
007155      000         IF((CMAX-CMIN).LE.1.0E-10)GO TO 90
007156      000         GO TO 105
007157      000   90    GO TO 101
007158      000   C*       PRINT FINAL SOLUTION
007159      000     100 CONTINUE
007160      000         WRITE(6,2100)CPCT
007161      000    2100 FORMAT(//,1X,F5.1,' % SOLUTION')
007162      000         CALL TIMOUT
007163      000         IF(.NOT.ALR) GO TO 101
007164      000         FOUND=.FALSE.
007165      000         ALR=.FALSE.
007166      000         CSAVE=1.0E38
007167      000         CPCT=0.
007168      000   C*       READ COST FACTOR FOR NEXT RUN
007169      000   101   READ(5,1000,END=900)CPCT1
007170      000         IF(CPCT1.LE.CPCT) GO TO 101
007171      000         CPCT=CPCT1
007172      000   102   CFACT=CPCT/100.0
007173      000         CMAX=CFACT*CSAVE
007174      000     105 IF(.NOT.FOUND)GO TO 120
007175      000         DO 110 I=1,NPTS
007176      000         IN(I)=.TRUE.
007177      000         IPOINT(I)=ISAVE(I)
007178      000         ACOST(I)=ASAVE(I)
007179      000         TCOST(I)=TSAVE(I)
007180      000         DO 110 J=1,NPTS
007181      000     110 EX(I,J)=EXSAVE(I,J)
007182      000         LAST=IPOINT(NPTS)
007183      000         NPTSIN=NPTS
007184      000         GO TO 40
007185      000     120 CONTINUE
007186      000         DO 130 I=1,NPTS
007187      000         DO 130 J=1,NPTS
007188      000     130 EX(I,J)=.FALSE.
007189      000         GO TO 5
007190      000   800   WRITE(6,3000)
007191      000   3000  FORMAT(//,' SOLVED ON ASCENT')
007192      000         CPCT=100.0
007193      000         CMIN=CMAX
007194      000         FOUND=.FALSE.
007195      000         GO TO 102
007196      000     900 CALL TIMOUT
007197      000         STOP
007198      000         END
007199      000   @ELT,SI TUMER-S*NONSYM.MAP-XQT,,,134275062312
007200      000   @MAP,S
007201      000   IN NONSYM.
007202      000   LIB SYSTEM$*MATHSTAT.
007203      000   END
007204      000   @XQT
007205      000   @ELT,SI TUMER-S*NONSYM.MAP-XQT,,,125316062112
007206      000   @MAP,N
007207      000   IN NONSYM.
```

```
007208      000   LIB SYSTEM$*MATHSTAT.
007209      000   END
007210      000   @XQT
007211      000   @ELT,SI TUMER-S*NONSYM.MINPT,,,1337081113111
007212      000         FUNCTION MINPT(I,N)
007213      000   C*      MINPT FINDS POINT WITH MINIMUM COST FOR CONNECTION TO
007214      000   C*      GIVEN POINT
007215      000         INCLUDE BLANK
007216      000         M=0
007217      000         C=1.0E38
007218      000         DO 10 J=1,NPTS
007219      000         IF(IN(J).OR.EX(N,J))GO TO 10
007220      000         IF(C.LE.COST(I,J))GO TO 10
007221      000         C=COST(I,J)
007222      000         M=J
007223      000      10 CONTINUE
007224      000         MINPT=M
007225      000         RETURN
007226      000         END
007227      000   @ELT,SI TUMER-S*NONSYM.MINPT2,,,1[illegible]0070112
007228      000         FUNCTION MINPT2(I)
007229      000   C*      MINPT2 FINDS POINT WITH [illegible]T COST FOR CONNECTION TO
007230      000   C*      GIVEN POINT
007231      000   C*        I=GIVEN POINT
007232      000         INCLUDE BLANK
007233      000         M=0
007234      000         C=1.0E38
007235      000         DO 10 J=1,NPTS
007236      000         IF(J.EQ.I.OR.IN(J)) GO TO 10
007237      000         IF(C.LE.COST(J,I)) GO TO 10
007238      000         C=COST(J,I)
007239      000         M=J
007240      000      10 CONTINUE
007241      000         MINPT2=M
007242      000         RETURN
007243      000         END
007244      000   @ELT,SI TUMER-S*NONSYM.ORDER/ORD,,,1763511121411
007245      000         SUBROUTINE ORDER(ISTART)
007246      000   C*      ORDER ORDERS LIST OF COMBINATIONS BY INCREASING COST
007247      000   C*      COMBINATIONS CONTAINING STARTING POINT ARE OMITTED
007248      000         INCLUDE BLANK
007249      000         N=NPTS-1
007250      000         NLIST=(N*N-N)/2
007251      000         IF(ISTART.EQ.0)NLIST=(N*NPTS)/2
007252      000         DO 30 K=1,NLIST
007253      000         C=1.0E38
007254      000         DO 20 I=1,N
007255      000         IF(I.EQ.ISTART)GO TO 20
007256      000         L=I+1
007257      000         DO 10 J=L,NPTS
007258      000         IF(J.EQ.ISTART.OR.EX(I,J))GO TO 10
007259      000         IF(C.LE.COST(I,J))GO TO 10
007260      000         C=COST(I,J)
007261      000         LIST(1,K)=I
007262      000         LIST(2,K)=J
007263      000      10 CONTINUE
007264      000      20 CONTINUE
```

```
007265  000          I=LIST(1,K)
007266  000          J=LIST(2,K)
007267  000       30 EX(I,J)=.TRUE.
007268  000          DO 40 I=1,NPTS
007269  000          DO 40 J=1,NPTS
007270  000       40 EX(I,J)=.FALSE.
007271  000          RETURN
007272  000          END
007273  000    @ELT,SI TUMER-S*NONSYM.OUTPUT/PATCH,,,110552062412
007274  000          SUBROUTINE OUTPUT(N,C,IPOINT)
007275  000          PARAMETER MAXPTS=60
007276  000    C*       OUTPUT PRINTS COST AND LIST OF POINTS
007277  000          DIMENSION IPOINT(MAXPTS)
007278  000          M=MIN(N,25)
007279  000          WRITE(6,2000)C,(IPOINT(I),I=1,M)
007280  000          IF(N.LE.25)GO TO 900
007281  000          DO 10 L=26,N,25
007282  000          M=MIN(N,L+24)
007283  000       10 WRITE(6,2100) (IPOINT(I),I=L,M)
007284  000     2000 FORMAT(' COST',F7.0,3X,'POINTS',25I4)
007285  000     2100 FORMAT(22X,25I4)
007286  000      900 RETURN
007287  000          END
007288  000    @ELT,SI TUMER-S*NONSYM.OUTPUT/RAND,,,172614113311
007289  000          SUBROUTINE OUTPUT(N,C,IPOINT)
007290  000    C*       OUTPUT PRINTS COST AND LIST OF POINTS
007291  000          DIMENSION IPOINT(1)
007292  000          M=MIN(N,12)
007293  000          WRITE(6,2000)C,(IPOINT(I),I=1,M)
007294  000          IF(N.LE.12)GO TO 900
007295  000          DO 10 L=13,N,12
007296  000          M=MIN(N,L+11)
007297  000       10 WRITE(6,2100) (IPOINT(I),I=L,M)
007298  000     2000 FORMAT(' COST',F7.0,3X,'POINTS',12I4)
007299  000     2100 FORMAT(22X,12I4)
007300  000      900 RETURN
007301  000          END
007302  000    @ELT,SI TUMER-S*NONSYM.RANDU,,,160784073712
007303  000    @ASG,A TUMER-S*NONSYM.
007304  000    @ASG,A TUMER-S*MATRIX.
007305  000    @USE 9,MATRIX.
007306  000    @ADD NONSYM.MAP-XQT
007307  000    30,0,1,0E30,15.,,30,3,0,0,0,0,1
007308  000    0,0,9
007309  000    5.,1.,,.2
007310  000    100.,3500.,1.111
007311  000    0.
007312  000    92.
007313  000    95.
007314  000    97.
007315  000    99.
007316  000    100.
007317  000    @EOF
007318  000    @ELT,SI TUMER-S*NONSYM.READIN,,,101664070312
007319  000    @ASG,A TUMER-S*NONSYM.
007320  000    @ASG,A FATHOM-DATA.
007321  000    @ASG,A TUMER-S*MATRIX.
```

```
007324      000   @USE 9,MATRIX.
007325      000   @ADD NONSYM.MAP-XQT
007326      000   6,0,1.0E30,0.,30,3,0,0,0,0,0
007327      000   0,0,9
007328      000   5.,1.,0.2
007329      000   @ADD FATHOM-DATA.NONSYM6
007330      000   95.
007331      000   97.
007332      000   99.9
007333      000   100.
007334      000   @EOF
007335      000   @ELT,SI TUMER-S*NONSYM.RFOR,,,[illegible]17063312
007336      000   @RFOR,CS NONSYM.TIMOUT
007337      000   @RFOR,CS NONSYM.IO
007338      000   @RFOR,CS NONSYM.COPY
007339      000   @RFOR,CS NONSYM.MINPT
007340      000   @RFOR,CS NONSYM.SHOW/BATCH,.[illegible]
007341      000   @RFOR,SC FATHOM.INPUT,NONSYM.INPUT
007342      000   @RFOR,CS NONSYM.MAIN
007343      000   @RFOR,CS NONSYM.OUTPUT/BATCH,.OUTPUT
007344      000   @RFOR,CS NONSYM.START
007345      000   @RFOR,CS NONSYM.CMINP3
007346      000   @RFOR,CS NONSYM.TREE
007347      000   @RFOR,CS NONSYM.INBAND
007348      000   @RFOR,CS NONSYM.MINPT2
007349      000   @EOF
007350      000   @ELT,SI TUMER-S*NONSYM.SHOW/BATCH,,,134704051012
007351      000         SUBROUTINE SHOW(NPTS,COST)
007352      000   C*       SHOW PRINTS COST MATRIX
007353      000         PARAMETER MAXPTS=60
007354      000         DIMENSION COST(MAXPTS,MAXPTS)
007355      000         WRITE(6,2000)
007356      000    2000 FORMAT(' COST MATRIX')
007357      000         DO 20 K=1,NPTS,16
007358      000         L=MIN(NPTS,K+15)
007359      000         WRITE(6,2100) (J,J=K,L)
007360      000         DO 20 I=1,NPTS
007361      000      20 WRITE(6,2200)I,(COST(I,J),J=K,L)
007362      000    2100 FORMAT(4X,16I7)
007363      000    2200 FORMAT(1X,I3,16F7.0)
007364      000         RETURN
007365      000         END
007366      000   @ELT,SI TUMER-S*NONSYM.SHOW/[illegible]AND,,,135036051012
007367      000         SUBROUTINE SHOW(NPTS,COST)
007368      000   C*       SHOW PRINTS COST MATRIX
007369      000         PARAMETER MAXPTS=60
007370      000         DIMENSION COST(MAXPTS,MAXPTS)
007371      000         WRITE(6,2000)
007372      000    2000 FORMAT(' COST MATRIX')
007373      000         DO 20 K=1,NPTS,9
007374      000         L=MIN(NPTS,K+8)
007375      000         WRITE(6,2100) (J,J=K,L)
007376      000         DO 20 I=1,NPTS
007377      000      20 WRITE(6,2200)I,(COST(I,J),J=K,L)
007378      000    2100 FORMAT(4X,9I7)
007379      000    2200 FORMAT(1X,I3,9F7.0)
007380      000         RETURN
```

```
      END
@ELT,SI TUMER-S*NONSYM.START
      SUBROUTINE START(
C*      START FINDS THE
      INCLUDE BLANK
      DIMENSION PISAVE(
      DIMENSION IPI(
     *,IPO(MAXPTS),ITEMP(
      LOGICAL PRINTM,PRINT(
C*      ORDER CONNECTIONS
      DO 5 I=1,NPTS
5     IN(I)=.FALSE.
      IF(ISTART.NE.0) GO TO 22
      CMAX=1.0E30
      DO 20 I=1,NPTS
      IN(I)=.TRUE.
      T=TREE(1,ITEMP,I
      C=CMINP3(I,K1,K2)
      C=C+T
      IF(C.LE.CMAX) GO TO 20
      ISTART=I
      CMAX=C
20    IN(I)=.FALSE.
22    IN(ISTART)=.TRUE.
      T=TREE(1,ITEMP,IT
      C=CMINP3(ISTART,K1,
      CMAX=C+T
      WRITE(6,2000) CMAX
2000  FORMAT(' COST',F10.
      ISAVE(1)=ISTART
C*      USE VOGEL METHOD TO
      IF(NPI.LE.0)GO TO
      CALL COPY(NPTS,COST,CT
      CALL TIMOUT
C*      ITERATE ON COST MATRIX TO FIND MAXIMUM
      JPI=1
      PI=PISAVE(1)
      WRITE(6,2100)PI
 2100 FORMAT(' PI=',F10.4)
      IT=0
23    J1=K1
      J2=K2
      DO 24 J=1,NPTS
      IPO(J)=ITEMPO(J)
24    IPI(J)=ITEMP(J)
C*      ADJUST COST
   25 IPI(ISTART)=0
      IPO(ISTART)=0
      IPI(J2)=IPI(J2)+1
      IPO(J1)=IPO(J1)+1
      ICOST=0
      DO 30 J=1,NPTS
      ICOST=ICOST+ABS(IPI(J))+ABS(IPO(J))
      DO 30 K=1,NPTS
30    COST(J,K)=COST(J,K)+PI*(IPO(J)+IPI(K))
      IF(ICOST.EQ.0) GO TO
      IF(PRINTM)CALL SHOW(NPTS,COST)
```

```
007436      000          IF(PRINTM) WRITE(6,2001) C
007437      000          IF(PRINTM) CALL TIMOUT
007438      000    C*      REORDER CONNECTIONS
007439      000    C*      COMPUTE MINIMUM COST
007440      000          T=TREE(1,IPI,IPO)
007441      000          C=CMINP3(ISTART,J1,J2)
007442      000          C=C+T
007443      000          IF(C.LE.CMAX)GO TO 40
007444      000    C*      NEW MAXIMUM FOUND
007445      000          IT=0
007446      000          CMAX=C
007447      000          CALL COPY(NPTS,COST,CTEMP)
007448      000          K1=J1
007449      000          K2=J2
007450      000          DO 35 J=1,NPTS
007451      000          ITEMP(J)=IPI(J)
007452      000    35    ITEMPO(J)=IPO(J)
007453      000          GO TO 25
007454      000       40 IT=IT+1
007455      000          IF(IT.LT.MAXIT)GO TO 25
007456      000    C*      GO BACK TO BEST COST
007457      000          CALL COPY(NPTS,CTEMP,COST)
007458      000          IF(IFOUT(JPI).NE.0)CALL ICOPY(IFOUT(JPI),1,NPTS,COST)
007459      000          WRITE(6,2001) CMAX
007460      000          CALL TIMOUT
007461      000          IF(JPI.EQ.NPI)GO TO 900
007462      000          IT=0
007463      000          JPI=JPI+1
007464      000          PI=PISAVE(JPI)
007465      000          WRITE(6,2100) PI
007466      000          GO TO 23
007467      000    800   FOUND=.TRUE.
007468      000    900   WRITE(6,2000) CSAVE,ISTART
007469      000          WRITE(6,2001) CMAX
007470      000    2001  FORMAT(' MAXIMUM COST',F7.0)
007471      000           CALL TIMOUT
007472      000          RETURN
007473      000          END
007474      000    @ELT,SI TUMER-S*NONSYM,TIMOUT,,,159574103711
007475      000          SUBROUTINE TIMOUT
007476      000          K=ITIME(I,J)
007477      000          T=I/5000.
007478      000          WRITE(6,2000)T
007479      000     2000 FORMAT(' TIME',F10.3,' SEC')
007480      000          RETURN
007481      000          END
007482      000    @ELT,SI TUMER-S*NONSYM,TREE,,,132152063412
007483      000          FUNCTION TREE(N,IPI,IPO)
007484      000          INCLUDE BLANK
007485      000          DIMENSION IPI(MAXPTS),IPO(MAXPTS)
007486      000          DIMENSION LINKI(MAXPTS),LINKO(MAXPTS),CLINK(MAXPTS)
007487      000          M=NPTS-N-1
007488      000          C=0.
007489      000          IF(M.LE.0) GO TO 900
007490      000    C*         FIND A ROW NOT IN INLIST AND INITIALIZE
007491      000          IROW=0
007492      000          DO 10 I=1,NPTS
```

```
                IPI(I)=-1
                IPO(I)=-1
                LINKO(I)=0
                IF(.NOT.IN(I)) IROW=I
   10           LINKI(I)=0
   C*               FIND LEAST COST ELEMENT IN ROW
                C=1.0E38
                DO 40 J=1,NPTS
                IF(IN(J).OR.J.EQ.IROW) GO TO 40
                IF(COST(IROW,J).GT.C) GO TO 30
                LINKI(J)=IROW
                CLINK(J)=COST(IROW,J)
                GO TO 35
   30           LINKO(J)=IROW
                CLINK(J)=COST(J,IROW)
   35           IF(CLINK(J).GE.C) GO TO 40
                ICOL=J
                C=CLINK(J)
   40           CONTINUE
   C*               SET NEW VECTORS FOR SHORT BRANCH
                IF(LINKI(ICOL).EQ.0) GO TO 45
                IPI(ICOL)=IPI(ICOL)+1
                IROW=LINKI(ICOL)
                IPO(IROW)=IPO(IROW)+1
                LINKI(ICOL)=0
                GO TO 50
   45           IPO(ICOL)=IPO(ICOL)+1
                IROW=LINKO(ICOL)
                IPI(IROW)=IPI(IROW)+1
                LINKO(ICOL)=0
   50           M=M-1
                IF(M.EQ.0) GO TO 900
   C*               ADD OTHER BRANCHES TO TREE
                DO 80 J=1,M
                CMIN=1.0E38
                DO 70 I=1,NPTS
                MES=LINKI(I)*LINKO(I)
                IF(MES.EQ.0) GO TO 70
                IF(CLINK(I).LE.COST(I,ICOL)) GO TO 60
                LINKI(I)=0
                LINKO(I)=ICOL
                CLINK(I)=COST(I,ICOL)
   60           IF(CLINK(I).LE.COST(ICOL,I)) GO TO 65
                LINKI(I)=ICOL
                LINKO(I)=0
                CLINK(I)=COST(ICOL,I)
   65           IF(CLINK(I).GE.CMIN) GO TO 70
                JCOL=I
                CMIN=CLINK(I)
   70           CONTINUE
                ICOL=JCOL
                IF(LINKI(ICOL).EQ.0) GO TO 75
                IPI(ICOL)=IPI(ICOL)+1
                IROW=LINKI(ICOL)
                IPO(IROW)=IPO(IROW)+1
                LINKI(ICOL)=0
                GO TO 80
```

```
007550      000    75     IPO(ICOL)=IPO(ICOL)+1
007551      000           IROW=LINKO(ICOL)
007552      000           IPI(IROW)=IPI(IROW)+1
007553      000           LINKO(ICOL)=0
007554      000    80     C=C+CLINK(ICOL)
007555      000    900    TREE=C
007556      000           RETURN
007557      000           END
007558      000    @PDP,FI TUMER-S*NONSYM.COMMON#/BLANK,,,156514061612
007559      000    BLANK   PROC
007560      000           PARAMETER MAXPTS=60,MXLIST=6000
007561      000           COMMON NPTS,NLIST,CLINK(MXLIST),LIST(2,MXLIST),
007562      000          *ISAVE(MAXPTS),COST(MAXPTS,MAXPTS),EX(MAXPTS,MAXPTS),
007563      000          *CMAX,FOUND
007564      000           LOGICAL IN,EX,FOUND
007565      000     END

END ELT.
@FIN


  RUNID: IERON    ACCT: 011A0936       PROJECT: PUBLIC

      IERON*MSG: CHG AND EXCESS LINES TO E-24-200

  TIME:    TOTAL: 00:00:33.017

           CPU:   00:00:09.476      I/O:  00:00:01.002

           CC/FR: 00:00:21.639      WAIT: 00:00:00.900

  IMAGES  READ:   7569      PAGES:  133

  START:    16:58:34 FEB 21,1975    FIN:  17:01:36 FEB 21,1975
```